Tischcomputer-Software

Anforderungen an Programme für professionelle Anwender

Unabhängig davon, ob man die für den individuellen Anwendungsfall erforderlichen Programme selbst schreibt oder schreiben läßt, muß man davon ausgehen, daß der spätere Computer-Benutzer kein Programmierfachmann ist. Dies führt zu einer Reihe von Forderungen und Empfehlungen für die Anwender-Software.

Tischcomputer werden im Gegensatz zu Großrechnern oder Minicomputern überall dort eingesetzt, wo vergleichsweise nur geringe Datenbestände anfallen, also in Kleinbetrieben oder auch bei dezentraler Datenverarbeitung in größeren Betrieben. Darunter fallen auch „Teillösungen", bei denen nicht ein einziger größerer Computer unterschiedlichste Aufgabenstellungen zu erfüllen hat, sondern bei denen für jede einzelne Aufgabenstellung ein eigener Tischcomputer vorhanden ist.

Situation beim Anwender

Der Anwender des Tischcomputers ist normalerweise nicht EDV-geschult. Er betrachtet den Computer nur als Mittel zum Zweck und möchte sich möglichst nicht um dessen Innenleben und Programmierung kümmern müssen.

In Kleinbetrieben sind die Investitionsmöglichkeiten meist nur relativ gering. Deshalb kommen in erster Linie preiswerte Tischcomputersysteme in Betracht (Hardwarekosten rund 10 000 DM). Die Softwarekosten müssen unbedingt in einem vernünftigen Verhältnis hierzu stehen; wir werden noch sehen, wie dieses Ziel erreicht werden kann.

Eine große Schwierigkeit ist oft, daß die Arbeitsabläufe recht „eingefahren" sind: Viele Büroangestellte haben sich z. B. an Papier-Karteien gewöhnt und möchten daran trotz aller Vorteile eines Computers freiwillig nichts ändern. Dazu kommt ein prinzipielles Mißtrauen gegen Neuerungen und gegen das mit ihnen verbundene Risiko.

Es ist interessant, daß dieses Problem bei den an sich sonst recht trägen Großbetrieben weniger auftaucht als bei den normalerweise flexibleren Kleinbetrieben.

Software-Grundforderungen

Wie schon erwähnt, darf man beim Anwender keine Vorkenntnisse voraussetzen. Folglich sollen alle systeminternen Dinge, wie z.B. Initialisieren von Peripheriegeräten, Laden von Programmen usw. möglichst automatisch ablaufen. Der Kontakt zwischen Mensch und Computer muß in einem leicht verständlichen Dialog erfolgen: Der Computer stellt eine Frage, bietet unterschiedliche Auswahlmöglichkeiten an und meldet sofort etwaige Bedienungsfehler, soweit sie syntaktisch erkennbar sind.

> **Es ist unmöglich, ein Programm idiotensicher zu machen, weil Idioten so erfindungsreich sind.**
> *E. Murphy*

Sinnvoll ist es auch, diesen Dialogverkehr zwar komfortabel zu machen, aber andererseits so aufzubauen, daß der Anwender ohne allzuviele Tastendrücke und möglichst zeitsparend die gewünschte Funktion ausführen kann.

Wie schon erwähnt, ist die Investitionsmöglichkeit von Kleinbetrieben nur beschränkt. Was für die Hardware gilt, muß auch für die Software gelten – sie darf nicht zu teuer sein. Sollten sich zu einem späteren Zeitpunkt die Anforderungen an ein bestimmtes Programm erweitern, so muß es sich an die neuen Verhältnisse leicht anpassen lassen, was automatisch auch eine gute Dokumentation aller Routinen bedingt.

Automatisierung des Ablaufs

Beim Einschalten des Computers steht das Anwenderprogramm normalerweise noch nicht im Arbeitspeicher, sondern muß erst (z.B. von einer Floppy-Disk) geladen werden. Das sollte möglichst einfach geschehen, z.B. durch Drücken einer Funktionstaste oder mit möglichst wenigen, leicht beherrschbaren Befehlen.

Ähnliches gilt für das Laden von Dateien sowie das Abspeichern nach etwaigen Änderungen, etwa bei einer Kundenkartei oder einem Lagerverzeichnis. Besteht ein Programm aus einem Initialisierungs- und einem Hauptteil, so sollte der Initialisierungsteil den Hauptteil vollautomatisch nachladen und möglichst auch gleich starten, wie das z.B. beim Texteditor-Programm in FUNKSCHAU 23/1980 realisiert ist.

Menütechnik

In den meisten Fällen wird ein und derselbe Computer für die unterschiedlichsten Aufgabenstellungen eingesetzt: so etwa für das Schreiben von Formbriefen, für das Führen einer Adressenkartei und für die Gehaltsabrechnung. Infolgedessen müssen je nach momentan gewünschtem Verwendungszweck unterschiedliche Programme geladen werden.

Hierzu hat sich die Menütechnik gut bewährt. Beim Einschalten des Computers oder bei Beendigung einer Aufgabe wird automatisch ein Menü-Programm geladen, das auf dem Bildschirm alle zur Verfügung stehenden Programme auflistet. Durch Drücken nur einer Taste (Kennbuchstabe oder Ziffer) kann der Benutzer dann das gewünschte Programm laden, ohne sich darum kümmern zu müssen, wo das jeweilige Programm auf der Floppy-Disk oder Kassette steht.

Denkbar sind auch hierarchische Menüs, die immer dann sinnvoll sind, wenn nicht alle Programme auf einmal auf den Schirm passen. Dabei wählt der Benutzer zunächst grob in einer ersten „Ebene" ganze Programmgruppen und in einer zweiten Ebene das genau gewünschte Programm. Für häufig gebrauchte Programme empfiehlt sich, den Zugriff direkt aus der ersten Ebene

zu ermöglichen. Aus jedem Programm sollte man ohne Schwierigkeiten zum übergeordneten Menü zurückgelangen können, ebenso aus einer tieferen Menü-Ebene.

Ausgefeilter Dialog

Zu einem guten Dialogverkehr gehören auch ergonomische Gesichtspunkte. Fragen und mögliche Antworten müssen klar und eindeutig fomuliert sein. Der Bildschirm ist von unnötigem Zierrat freizuhalten, wie etwa Sternchen, Herzchen usw.; Rahmen sind nur zur Zusammenfassung von bestimmten Bildschirmteilen sinnvoll. Invers-Felder (d. h. Felder, auf denen schwarze Schrift auf weißem bzw. grünem Hintergrund dargestellt wird) sind wegen der erhöhten Augenbelastung möglichst zu vermeiden, soweit sie nicht wirklich der Übersichtlichkeit dienen.

Diejenige Stelle auf dem Bildschirm, auf der das nächste Zeichen geschrieben wird, das der Benutzer eingibt, wird üblicherweise vom „Cursor" markiert. Er kann blinken oder statisch vorhanden sein (helles Quadrat). Die Auswahl sollte man dem Anwender überlassen, auch die Blinkgeschwindigkeit sollte er seinen eigenen Bedürfnissen anpassen können, z. B. in drei Stufen.

Bildschirmaufteilung

Für die Übersichtlichkeit der dargestellten Informationen ist es nützlich, wenn man das Durchrollen („Scrolling") vermeidet. Darunter versteht man, daß der gesamte Bildschirminhalt um eine Zeile nach oben rückt, wenn der Cursor am unteren Bildrand angelangt ist.

Besser ist es, für bestimmte Bearbeitungsabschnitte den Bildschirm vollständig zu löschen und oben neu zu beginnen. Eine Ausnahme stellt natürlich die Bearbeitung längerer Texte dar. Zusammengehörige Information soll unbedingt gleichzeitig auf dem Schirm stehen.

Sehr nützlich ist die Möglichkeit zum „Screen Editing": Der Bildschirm dient sozusagen als Formular, und der Anwender kann mit dem Cursor an die jeweiligen Felder gehen, um dort etwas einzugeben oder zu korrigieren.

Bei vielen Programmen (z.B. Texteditor) ist es sinnvoll, am oberen oder unteren Bildrand eine „Systemzeile" darzustellen, die während des gesamten Programmablaufs beibehalten wird und alle aktuellen Parameter (z. B. Zeilenzahl und -länge, freier Speicherplatz usw.) darstellt.

Fehlerkorrektur und -behandlung

Manche Programmierer sind der Ansicht, daß sich der Anwender gefälligst selbst darum kümmern sollte, daß er keine Fehler bei der Eingabe macht – dies sei ein nicht unerwünschter Erziehungseffekt.

In der Praxis hat sich das allerdings nicht bewährt: Wegen des damit auftretenden Streß werden eher mehr Fehler gemacht. Also muß das Programm über möglichst komfortable Korrekturmöglichkeiten verfügen, auch für Informationen, die mit der „Return"-Taste schon in den Arbeitsspeicher übernommen wurden.

Eine deutliche Reduktion der Fehlerwahrscheinlichkeit bei der Eingabe bewirken Plausibilitätskontrollen durch das Programm: Unzulässige Zeichen müssen ignoriert werden, z.B. die Buchstabentasten bei einer Zahleneingabe; eine zu kurze oder zu lange Eingabe darf nicht zugelassen werden und muß zu einer Fehlermeldung sowie zur Neueingabe führen. Auch inhaltliche Fehler müssen zurückgewiesen werden, beispielsweise, wenn der Benutzer für sein Geburtsdatum 54321.98 eingibt. Dagegen sollte der Computer wissen, was gemeint ist, wenn 12,3,50 oder 12,3,1950 eingetippt wird. Unsinnig und fehlerträchtig wäre es, wenn der Computer das Geburtsdatum in einer ungewohnten Reihenfolge verlangt, etwa als „Jahr, Monat, Tag".

Funktionssicherheit der Software

Der zuverlässige Ablauf eines Programms beginnt schon bei der Verwendung geeigneter Eingaberoutinen. Bei manchen Computern (PET, CBM, PC-100, AIM-65) führt das Drücken der Return-Taste ohne weitere Eingabe bei einem INPUT-Befehl zu einem Programmabbruch, und die meisten Benutzer würden darauf alles andere tun, als CONT zur Weiterführung einzugeben.

Infolgedessen ist es empfehlenswert, eigene Routinen für die Eingabe zu verwenden, die dieses Problem umgehen.

Die Dateibehandlung sollte weitgehend automatisiert sein. Wird z.B. eine nicht initialisierte Diskette eingelegt, so sollte dies der Computer selbst durch Auswerten der Fehlermeldung erkennen und die Initialisierung vornehmen. Bei anderen Systemfehlern sollten Korrekturmöglichkeiten vorhanden sein, ohne daß das Programm abgebrochen wird. Entsprechende Programmabschnitte sollten auch das Auftreten unzulässiger Werte bei arithmetischen Operationen erkennen, damit z.B. bei einer Division durch Null kein Programmabbruch und Datenverlust auftritt.

Zur Funktionssicherheit gehört aber auch die Zuverlässigkeit des Programms selbst. Viele Fehler stellen sich erst nach langer Betriebszeit heraus. Da viele Computer bei der Eingabe syntaktisch falscher Basic-Programmzeilen keine Fehlermeldung abgeben, sondern dies erst bei der Ausführung der entsprechenden Zeile tun, stellen sich manche Fehler auch erst bei bestimmten, vielleicht unwahrscheinlichen Betriebssituationen heraus, die normalerweise z. B. durch eine IF...THEN-Anweisung umgangen werden.

> **Es ist immer noch ein Fehler vorhanden.**
> *E. Murphy*

Es sollte selbstverständlich sein, daß dem praktischen Einsatz eines Programms umfassende Tests vorausgehen, auch durch Laien. Möglicherweise ist auch ein Piloteinsatz „außer Konkurrenz" als erster Praxistest sinnvoll.

Eine Selbstverständlichkeit sollte eigentlich eine umfassene, systematisch aufgebaute und für den Anwender verständliche Bedienungsanleitung sein. Die Programmdokumentation muß so ausführlich sein, daß auch ein Programmierer, der das Programm zum ersten Mal sieht, ohne Schwierigkeiten Änderungen, Anpassungen und eventuell Verbesserungen vornehmen kann.

Besonders bewährt hat sich eine besondere Kurzanleitung für den Anwender. Sie versetzt ihn in die Lage, die Inbetriebnahme in der von seiner Neugier erzwungenen Schnelligkeit vorzunehmen. Die Kurzanleitung soll nur die wichtigsten Schritte enthalten, die für das Umgehen mit dem Programm nötig sind.

Rationeller Programmablauf

Tischcomputer sind kein Ersatz für die Mittlere Datentechnik oder gar für Großrechner. Die Forderung nach einem rationellen Programmablauf konzentriert sich deshalb vorwiegend auf ergonomische Gesichtspunkte.

Sind etwa in einem Programm längere Rechenzeiten zu erwarten, so sollte man das Programm in einen Eingabe-, einen Rechen- und einen Ausgabeblock

trennen, um unnötige Wartezeiten des Anwenders zu vermeiden. Überhaupt ist während der Eingabephase auf eine möglichst geringe Antwortzeit des Rechners zu achten. Selbstverständlich lassen sich auch die Rechenzeiten oft noch durch einen durchdachten Programmaufbau reduzieren. Der Einsatz von Universalroutinen aus irgendwelchen Quellen ist meist nicht optimal, wenn auch billig.

Parameter-Dateien

Zahlreiche Systemparameter, wie z.B. Drucker-Zeilenbreite, bevorzugte Arbeitsmodi, Cursorart usw. sollten vom Benutzer nur einmal eingegeben werden müssen, nämlich bei der ersten Inbetriebnahme des Programms, wenngleich spätere Änderungen dieser Parameter möglich sind. Dadurch entfallen lästige Routineabfragen. Ideal ist die Schaffung einer speziellen Parameter-Datei auf der Diskette, die mit einem mitgelieferten Generatorprogramm erzeugt wird. Dieses Generatorprogramm kann der Benutzer bei Bedarf laden und seine Systemparameter eingeben bzw. korrigieren. Das eigentliche Anwenderprogramm holt sich dann beim Start jeweils automatisch die Parameter aus dieser Datei von der Diskette.

Geringe Software-Kosten

Typische Tischcomputer-Systeme kosten heute (mit den nötigen Peripheriegeräten wie Drucker und Floppy-Disk) um 10 000 DM. Die Software-Entwicklungskosten müssen in einem vernünftigen Verhältnis hierzu stehen.

Ausgefeilte Software zu vernünftigen Preisen anbieten zu können, erfordert Standardprogramme mit hoher Auflage. Die meisten Programme sollten aus Standard-Moduln zusammengesetzt sein, die sehr ausgefeilt sind.

Jedes Programm darf nicht ausschließlich auf die Bedürfnisse eines Einzelbetriebes mit all seinen individuellen Forderungen zugeschnitten sein, sondern sollte die Anforderungen aller ähnlich gelagerten Betriebe abdecken. Individuelle Forderungen können dann z.B. mit den oben erwähnten Parameter-Dateien vom Benutzer selbst programmiert werden.

Für den Software-Hersteller bedeutet dies: Genaue Marktanalysen, Integration aller typischen Anforderungen, Verzicht auf Nebensächlichkeiten im Interesse der Überschaubarkeit der Software, sinnvolle und vor allem auch mit anderen Programmen kompatible Dateistrukturen und, nicht zuletzt, ein möglichst breit gefächerter Vertriebsweg für die Software.

Hans Beck, Herwig Feichtinger

µC-Nachrichten

Siemens brachte vor kurzem zwei sehr gut gestaltete Handbücher für den Tischcomputer **PC-100** heraus: Das „Basic-Handbuch" und das Monitor-Editor-Betriebssystem"-Listing. Vor allem letzteres – den AIM-65-Besitzern wird es schon bekannt sein – sollte anderen Mikrocomputer-Herstellern als Vorbild dienen. Das Basic-Handbuch enthält eine genaue Aufstellung der möglichen Befehle und deren Wirkung, auch im Vergleich mit anderen Basic-Dialekten.

F'

Commodore plant, seine **CBM-Computer** endlich auch für die Textverarbeitung in Deutschland brauchbar zu machen: Spätestens bis Ende nächsten Jahres wird eine Version mit den Umlauten ä, ö, ü erhältlich sein. Außerdem wird geplant, einen neuen, transportablen Computer zu bauen, der mit der CMOS-Version des Mikroprozessors 6502 und einem LCD-Matrix-Display arbeiten soll.

Nascom-Verbesserung

Bei dem Mikrocomputer Nascom-1 entstehen während des Schreibvorganges auf dem Bildschirm wild verlaufende Linien und Punkte, weil dann das Video-RAM gleichzeitig ausgelesen und beschrieben wird. Zu diesen Störungen kommt es, weil in das Video-RAM beim Schreibvorgang gleichzeitig gelesen und geschrieben wird.

Um den Bildschirm sauber zu bekommen, wird ein Monoflop mit dem $\overline{\text{VDUSEL}}$-Signal getriggert. Während der Verzögerungszeit (ca. 1 µs) wird der Bildschirm dunkelgesteuert (*Bild 1*). Jetzt ist bei jedem Schreibvorgang der Bildschirm für 1 µs dunkel und die störenden Linien werden unsichtbar.

Der Aufbau erfolgt am besten auf einer kleinen Lochrasterplatte, auf der man den 74 LS 122 und eine Wire-Wrap Fassung für den 74 LS 165 einlötet, welche ihrerseits wieder in die alte Fassung für den 74 LS 165 gesteckt wird. Pin 15 läßt man aus der Fassung herausragen und verbindet ihn mit dem Monoflop. Dadurch ist die Verdrahtung der Erweiterung auf die Verbindung Monoflop – $\overline{\text{VDUSEL}}$, also einen Draht beschränkt (*Bild 2*). Kurt Moraw

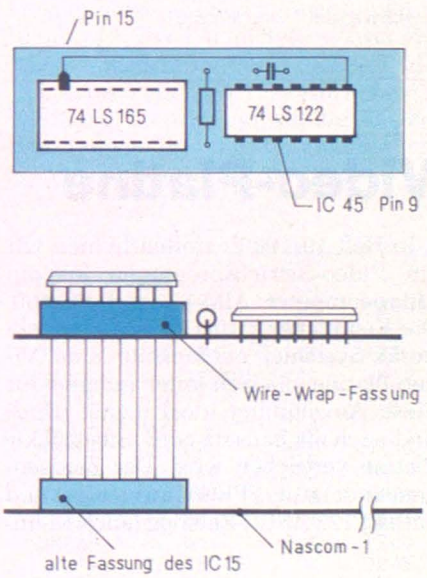


Bild 2. Einbau der Zusatzschaltung in die Nascom-Platine

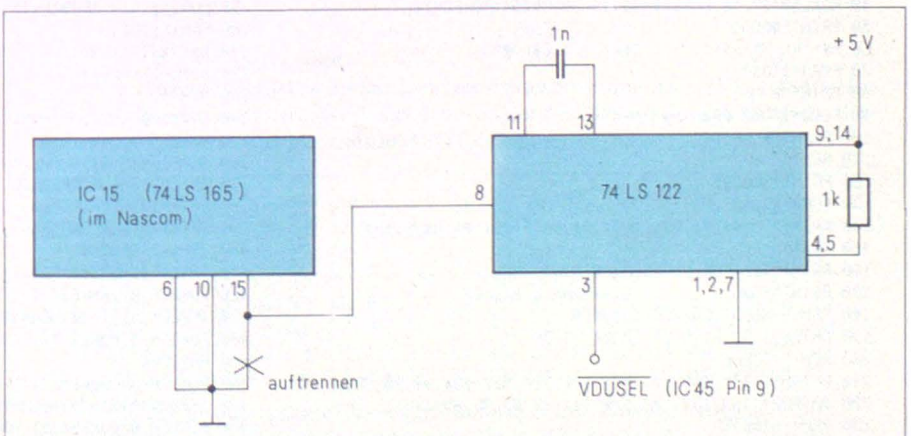
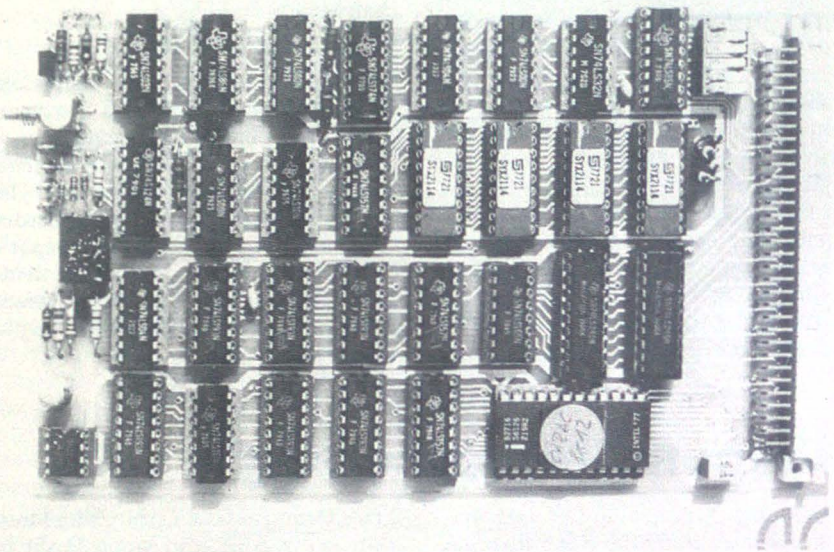


Bild 1. Zur Unterdrückung des Bildschirmflimmerns beim Schreiben genügt ein Monoflop



Video-Platine für direkten Speicherzugriff von GES. Die-Verbindung mit dem Mikrocomputer geschieht hier über Adressen- und Datenbus, so daß kein I/O-Port „verschwendet" werden muß

PET als Klavier

Mit dem im Mikrocomputer PET eingebauten User-VIA 6522, die unter anderem auch zwei programmierbare 16-bit-Timer enthält, lassen sich mit geringem Software-Aufwand beliebige Töne erzeugen, ohne dafür umständlich Schleifen programmieren zu müssen.

Das in *Bild 1* aufgelistete Klavier-Programm macht davon Gebrauch und liefert ein einstimmiges Nf-Ausgangssignal direkt am Anschluß CB2 des User-VIA. Es stellt auf dem PET-Bildschirm eine Klaviertastatur in Form weißer und schwarzer Balken dar *(Bild 2)*, und die Betätigung irgendeiner Taste erzeugt einen Ton, wobei auf dem Bildschirm an entsprechender Stelle ein Punkt eingeblendet wird, um eine Orientierung zu ermöglichen.

Das Programmlisting enthält einige PEEK- und POKE-Befehle, die zur Steuerung des User-VIA dienen und auch die Tastatur abfragen.

W. Gietmann, W. Billen

Video-Platine

In Heft 19/1980 veröffentlichten wir ein Video-Betriebsprogramm für die Mikrocomputer AIM-65 und PC-100. Die Kemptener Firma GES (Graf Elektronik-Systeme) entwickelte eine Video-Platine, die sich unter anderem für diese Anwendung ideal eignet (*Bild*) und auch als Bausatz oder unbestückte Platine vertrieben wird. Der Zeichengenerator ist als EPROM ausgeführt und enthält 128 ASCII-Zeichen (auch Kleinschreibung und Grafik-Symbole). Die mitgelieferte Beschreibung stellt den Anschluß an unterschiedliche Mikrocomputer-Systeme ausführlich dar.

Die Verbindung mit dem Mikrocomputer erfolgt über den Adressen- und Datenbus: Jede Zeichenposition auf dem Schirm entspricht einer bestimmten Adresse. Die Zeichen werden als 8 x 12-Punktmatrix dargestellt, d. h. mit Unterlängen.

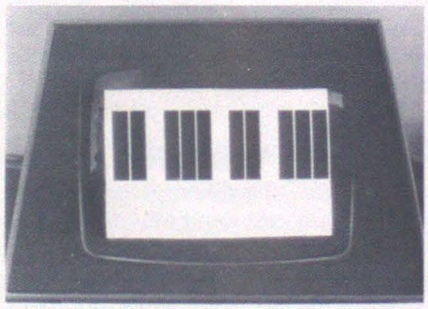


Bild 2. Auf dem Bildschirm des PET wird übersichtlich eine Klaviertastatur dargestellt, wobei die jeweils gedrückte Taste mit einem Punkt markiert ist (Foto: Fe.)

```
10 REM ***KLAVIER***
20 POKE59409,52
30 DIMP(24),S%(24),T%(24),A%(255),B(255)
40 POKE59467,16:POKE59466,15:CA=59464:POKECA,0
50 PRINT"♥QQQQ"
60 A$="R  |":S$="_  |":B$="   ":C$="■"
70 FORJ=1TO18
80 PRINT"R |";
90 FORI=1TO12:PRINTA$;:NEXT
100 PRINT"R  ";
110 NEXTJ
120 PRINT"SQQQQ"
130 FORJ=1TO10
140 PRINT" ";S$;S$;B$;S$;S$;S$;B$;S$;S$;B$;S$;S$;S$;
150 NEXTJ
160 PRINT"SQ":FORI=1TO120:PRINTC$;:NEXTI
170 PRINT"SQQQ        R GIETMANN & BILLEN "
180 REM---HALBTOENE/POSITIONEN
190 DATA0,3,9,12,15,21,24,30,33,36
200 REM---TOENE
210 DATA237,211,188,177,157,140,124,117,104,93,88,78,69,62
220 DATA224,199,167,149,132,111,99,83,73,65
230 REM---TASTEN
240 DATA64,56,63,55,62,54,61,53,60,52,59,58,50,57
250 DATA72,79,78,70,77,76,68,74,66,73
260 FORI=1TO14:P(I)=33525+I*3:NEXTI
270 FORI=15TO24:READH:P(I)=H+33250:NEXT
280 FORI=1TO24:READS%(I):NEXT
290 FORI=1TO24:READT%(I):NEXT
300 FORI=1TO24
310 A%(T%(I))=S%(I)
320 B(T%(I))=P(I)
330 NEXTI
340 GV%=160:GN%=209:HV%=32:HN%=81
350 POKE59409,60
360 T%=PEEK(515):H%=A%(T%):IFT%=6THEN500
370 IFT%=26THENPOKE59466,15
380 IFT%=18THENPOKE59466,51
390 IFT%=25THENPOKE59466,85
400 IFH%=0THEN360
410 IFT%)64THEN460:REM---HALBTOENE
420 POKECA,H%:POKEB(T%),GN%
430 IFPEEK(515)=T%THEN430
440 POKECA,0:POKEB(T%),GV%
450 GOTO360
460 POKECA,H%:POKEB(T%),HN%
470 IFPEEK(515)=T%THEN470
480 POKECA,0:POKEB(T%),HV%
490 GOTO360
500 POKE59467,0
```

Bild 1. Vollständiges Programmlisting des Klavier-Programms für den PET. Auf anderen Computern wie etwa dem TRS-80, die keinen programmierbaren Timer enthalten, ist es leider nicht verwendbar

Eduard J. Simon

KIM-Routinen für Schreibmaschine

In Heft 4/1980 veröffentlichten wir einen Schreibmaschinen-Zusatz, der elektromechanisch alle Tasten betätigen kann. Hier folgt nun eine Reihe nützlicher Routinen, um dieses Hardware-Interface an einem Mikrocomputer KIM-1 betreiben zu können. Für das Umschreiben auf andere 6502-Systeme empfiehlt sich die Zuhilfenahme der Systemadressen-Tabelle in unserem Sonderheft „Mikrocomputer-Anwendungen".

Um das in Heft 4/1980 beschriebene Interface (Schreibmaschine wird zum Schönschreibdrucker) zusammen mit einem Terminal zu verwenden, das an die serielle Schnittstelle des KIM-1 angeschlossen ist, benötigt die im gleichen Heft angegebene „6502-Routine zur Schreibmaschinen-Ansteuerung" eine Reihe von Ergänzungen (*Bilder 1...7*).

Die im KIM-1-Monitor vorhandenen Befehle, wie OUTCH ($1EA0_{16}$) oder CRLF ($1E2F_{16}$) haben den Nachteil, daß bei der Rückkehr die Registerinhalte zerstört sind. Um diese Routinen sowohl für das Video-Interface als auch für die Parallelausgabe zum Drucker zu verwenden ist es notwendig, auch Y-Register und Akku jeweils vor dem Einsprung in die serielle Routine zu retten.

Der für die Ansteuerung des Druckers vorgesehene Parallelausgang verwendet nur die Anschlüsse PA0 bis PA5; PA6 blieb unbenützt. Es bietet sich an, diese Leitung als Eingang zu verwenden, der abgefragt werden kann, ob der Drucker in Funktion treten oder nicht angesteuert werden soll. Ein einfacher Schalter, mit dem die Leitung PA6 mit Masse verbunden werden kann (Drukker ein), gestattet diese Auswahl der Funktionen; die Abfrage erfolgt im Programm mit dem BIT-Befehl.

In dem beiliegenden Programm, das leicht verschoben werden kann, wurde die in Heft 4/1980 vorgeschlagene Routine nur insofern geändert, daß das einfache Testprogramm zum Schreiben auf der Maschine vom ASCII-Terminal aus an den Anfang der Routine gestellt wurde; das restliche Programm selbst und die Tabelle für die Zeichenumwandlung wurden um hex 10 Zellen verschoben, und die Codetabelle entspricht der Verwendung mit einer „Olivetti lettera 36", die preisgünstig zu erhalten ist.

```
DE00-  20 5A 1E   JSR   1E5A      DE2C-  68        PLA
DE03-  20 0B DE   JSR   DE0B      DE2D-  A2 0A     LDX   #0A
DE06-  4C 00 DE   JMP   DE00      DE2F-  48        PHA
DE09-  EA         NOP             DE30-  29 3F     AND   #3F
DE0A-  EA         NOP             DE32-  20 3E DE  JSR   DE3E
DE0B-  A0 BF      LDY   #BF       DE35-  A2 14     LDX   #14
DE0D-  8C 01 17   STY   1701      DE37-  68        PLA
DE10-  86 F4      STX   F4        DE38-  10 02     BPL   DE3C
DE12-  AA         TAX             DE3A-  A2 60     LDX   #60
DE13-  BD 50 DE   LDA   DE50,X    DE3C-  A9 FF     LDA   #FF
DE16-  48         PHA             DE3E-  A0 00     LDY   #00
DE17-  29 40      AND   #40       DE40-  8D 00 17  STA   1700
DE19-  C5 F5      CMP   F5        DE43-  20 4E DE  JSR   DE4E
DE1B-  F0 0F      BEQ   DE2C      DE46-  88        DEY
DE1D-  A6 F5      LDX   F5        DE47-  D0 F7     BNE   DE40
DE1F-  85 F5      STA   F5        DE49-  CA        DEX
DE21-  F0 04      BEQ   DE27      DE4A-  D0 F4     BNE   DE40
DE23-  A9 36      LDA   #36       DE4C-  A6 F4     LDX   F4
DE25-  D0 02      BNE   DE29      DE4E-  60        RTS
DE27-  A9 33      LDA   #33       DE4F-  00        BRK
DE29-  20 2D DE   JSR   DE2D
```

Bild 1. Modifizierte 6502-Schreibmaschinen-Druckerroutine nach Heft 4/1980. Hex DE00 ist die Startadresse für das Schreiben von der ASCII-Tastatur, hex DE0B die Einsprungroutine zum Ausdruck eines ASCII-Zeichens

```
DE50   3F 3F 3F 3F 3F 3F 3F 3F C6 34 3F 3F 3F B5 3F 3F
DE60   3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F
DE70   34 69 70 6C 41 60 6B 69 58 6A 6C 42 0C 04 03 48
DE80   09 0D 30 2C 20 2B 18 2A 10 29 45 0C 0A 44 1D 4C
DE90   00 6E 53 5B 65 61 5A 5D 52 56 55 4A 4D 4B 54 49
DEA0   4E 72 66 62 59 51 5C 6D 64 63 5E 58 48 6A 40 50
DEB0   34 2E 13 1B 25 21 1A 1D 12 16 15 0A 0D 0B 14 09
DEC0   0E 32 26 22 19 11 1C 2D 24 23 1E 58 48 6A 01 C6
```

Bild 2. Umcodiertabelle für eine Schreibmaschine „Olivetti lettera 36". Die Tabelle unterscheidet sich etwas von der in Heft 4/1980 für die „Gabriele 2000" angegebene

```
DE0D-  8C 01 17   STY   1701
DE10-  86 F4      STX   F4
DE12-  AA         TAX
DE13-  BD 50 DE   LDA   DE50,X
DE16-  48         PHA
DE17-  29 40      AND   #40
DE19-  C5 F5      CMP   F5
DE1B-  F0 0F      BEQ   DE2C
DE1D-  A6 F5      LDX   F5
DE1F-  85 F5      STA   F5
DE21-  F0 04      BEQ   DE27
DE23-  A9 36      LDA   #36
DE25-  D0 02      BNE   DE29
DE27-  A9 33      LDA   #33
DE29-  20 2D DE   JSR   DE2D
```

Bild 3. Routine für den Schreibmaschinen-Druck entsprechend "OUTCH" im KIM-Monitorprogramm

Für die Funktionen OUTCH (ein ASCII-Zeichen ausdrucken), OUTSP (einen Leerraum ausdrucken) und CRLF (Wagenrücklauf und Zeilenvorschub) reicht der restliche Raum der für die Druckerroutine vorgesehenen Seite aus (im Programm hex DE_{16}).

Für verschiedene Programme ist es jedoch notwendig auch über die Funktionen PRTBYT (ein Byte – zwei Zeichen – ausdrucken) und PRTPNT (den Inhalt der Zellen FB, FA ausdrucken) zu verfügen, hierfür sind weitere 30_{16} Zellen notwendig, die Programme entsprechen dem KIM-1-Monitor.

```
DEE6-   2o 9E 1E   JSR   1E9E
DEE9-   A9 2o      LDA   *2o
DEEB-   85 Eo      STA   Eo
DEED-   2o D7 DE   JSR   DED7
DEFo-   6o         RTS
```

Bild 4. "OUTSP" im KIM-Monitor druckt einen Leerraum; diese Routine tut dies auf der Schreibmaschine

```
DEF1-   2o 2F 1E   JSR   1E2F
DEF4-   A9 oD      LDA   *oD
DEF6-   85 Eo      STA   Eo
DEF8-   2o D7 DE   JSR   DED7
DEFB-   A5 Eo      LDA   Eo
DEFD-   A4 E2      LDY   E2
DEFF-   6o         RTS
```

Bild 5. "CRLF"-Routine für den Schreibmaschinen-Ausdruck: Es wird ein Wagenrücklauf erzeugt, so daß eine Zeile begonnen wird

```
DFoo-   85 FC      STA   FC
DFo2-   4A         LSR   A
DFo3-   4A         LSR   A
DFo4-   4A         LSR   A
DFo5-   4A         LSR   A
DFo6-   2o 11 DF   JSR   DF11
DFo9-   A5 FC      LDA   FC
DFoB-   2o 11 DF   JSR   DF11
DFoE-   A5 FC      LDA   FC
DF1o-   6o         RTS
DF11-   29 oF      AND   *oF
DF13-   C9 oA      CMP   *oA
DF15-   18         CLC
DF16-   3o o2      BMI   DF1A
DF18-   69 o7      ADC   *o7
DF1A-   69 3o      ADC   *3o
DF1C-   2o Do DE   JSR   DEDo
DF1F-   6o         RTS
```

Bild 6. "PRTBYT" druckt ein Byte als Folge zweier ASCII-Zeichen hexadezimal aus

```
DF2o-   A5 FB      LDA   FB
DF22-   2o oo DF   JSR   DFoo
DF25-   2o 91 1F   JSR   1F91
DF28-   A5 FA      LDA   FA
DF2A-   2o oo DF   JSR   DFoo
DF2D-   2o 91 1F   JSR   1F91
DF3o-   6o         RTS
DF31-   oo         BRK
```

Bild 7. "PRTPNT" druckt den Inhalt des Adressen-Zeigers 00FA, 00FB in Form von vier hexadezimalen ASCII-Zeichen

Der auf dieser Seite verbleibenden Raum kann z. B. dazu verwendet werden dort das Programm „Textausdruck mit Baudot-Fernschreiber" aus Heft 16/1979 der Funkschau unterzubringen das, mit einer geringen Änderung – Einsprung in die Druckroutine nicht mit 20 00 01, sondern mit 20 D0 xx – den Textausdruck sowohl auf dem Bildschirm allein als auch zusammen mit dem Drucker erlaubt. Auch andere in der FUNKSCHAU veröffentlichte Programme können durch Änderung der Routinen entsprechend angepaßt werden, wie z. B. der im Sonderheft „Programme für Kleincomputer und Taschenrechner" beschriebene Texteditor. Hier ist es besonders von Vorteil, die Editierung nur am Bildschirm ausführen zu können, bei Überschreiten der auf dem Schirm verfügbaren Seite den bereits redigierten Text durch den Schalter in Leitung PA6 jedoch auszudrucken.

Meine Meinung...

Wurzeln mit der Beißzange gezogen

Wenn sich jemand auf komplizierte Weise einer Aufgabe unterzieht, die auch auf einfache Art gelöst werden könnte, dann sagt man in einigen deutschen Landschaften gern plastisch dazu: Der zieht die Hose mit der Beißzange an. Daß man durchaus – in diesem Sinne – auch mit der Beißzange programmieren kann, dies zeigte unlängst eine bundesdeutsche „Zeitschrift für Mikrocomputertechnik". Da führte ein Autor nämlich Klage darüber, daß man in den meisten Basic-Dialekten einschließlich der Microsoft-Varianten nicht unmittelbar Wurzeln höherer Ordnung ermitteln könne, und stellte dann über zwei Druckseiten hinweg ein kompliziertes Näherungsverfahren für eben diese Aufgabe vor. Alles dies schlüssig vorgetragen, mit Schaubildern dokumentiert und durch Ausflüge in die Geschichte der Mathematik anschaulich erläutert, und dennoch unnötig wie ein Kropf, denn selbst schlichtere Basic-Interpreter können sehr wohl derlei Wurzeln auf simple Weise errechnen: Man bildet einen Logarithmus des Radikanden, teilt ihn durch die Ordnungszahl der Wurzel und verwandelt zurück – c'est tout.

Nichts liegt uns nun ferner, als uns über ein Versehen zu mokieren, wie es jederzeit selbst den gewissenhaftesten Autoren und den fachkundigsten Redaktionen unterlaufen kann. Die erwähnte Beißzangen-Methode zum Wurzelziehen gibt vielmehr Anlaß, über die Lückenhaftigkeit des Dokumentationsmaterials zu den munter verkauften Mikrocomputern Klage zu führen. Selbst ansonsten hochgelobte Handbücher lassen den Programmierer hier allein. Wir haben den konkreten Fall „Wurzeln höherer Ordnung" zum Anlaß genommen, bei einigen Rechnerherstellern zu erfragen, warum man dem Benutzer nicht Besseres mit auf den Weg gebe. Sofern die Firmen darauf verweisen, der Ausbau der Dokumentationsschriften werde Schritt für Schritt weiter fortgeführt, bedarf die Antwort keines weiteren Kommentars.

In zwei Fällen jedoch kamen Antworten, die sehr wohl eine nähere Betrachtung verlangen. Einmal hieß es, wenn man einen Flügel kaufe, verlange man vom Instrumentenhersteller ja auch kein Buch „Wie spiele ich richtig Chopin". Im zweiten Fall sagte jemand, Wurzelziehen durch Logarithmus-Teilen gehöre zum mathematischen Grundwissen, und wer das nicht habe, der solle die Finger von Mathematik-Programmen lassen.

Beide Äußerungen hält der Verfasser, dies mit Verlaub gesagt, für arrogant, jede auf ihre Weise; sie zeigen darüber hinaus, daß manche Leute noch immer nicht verstanden haben, wohin die Entwicklung der Mikrocomputertechnik zielt: Von einer Disziplin, die anfänglich von einer Kaste hochspezialisierter Profis und genial-schrulliger Autodidakten getragen wurde, zur Massentechnologie. Dies aber bedeutet nichts anderes, als daß es nun allmählich doch höchste Zeit wird, Bedienungshandbücher dem Benutzerkreis anzupassen. Mikrocomputer-Hersteller machen nämlich bereits heute den Löwenanteil ihres Umsatzes nicht mit Leuten, die ihre Programme mit der Genialität eines Klaviervirtuosen komponieren oder auf einem Oberstufen-Leistungskurs in Mathematik aufbauen können, sondern mit Technikern und Hobbyisten weitaus schlichteren Zuschnitts. In dieser Situation erweisen sich Querverweise in die schöne Kunst der Klavierbedienung oder Bezugnahmen auf mathematisches Spezialisten-„Grundwissen" als wenig hilfreich. Was not tut, sind ordentliche Handbücher mit ausreichenden Angaben, die es dem Benutzer ersparen, sich beim Programmieren der Beißzange bedienen zu müssen. Hans-G. Joepgen

PET-Menue

Wer hat sich nicht schon einmal über die langwierige Sucherei beim Laden von Programmen von der PET-Kassette geärgert? Da der nachträgliche Einbau eines Bandzählwerks nicht jedem liegt und auch die angebotenen 2x5-Minuten-Kassetten keine echte Alternative bieten, bleibt praktisch nur noch die „Menü-Technik".

Bei dieser Technik enthält das erste Programm auf der Kassette alle nötigen Informationen für die Auswahl eines der restlichen Programme sowie ein Suchprogramm, welches den Benutzer im Dialogbetrieb an die gewünschte Stelle führt.

Da der Recordermotor des PET durch die Software ein- und ausgeschaltet werden kann, ist das hier vorliegende Programm, welches die gewünschte Bandstelle sucht, relativ kurz. Die Zeit, die man vom Ende dieses Programmes bis zum gewünschten Programm vorspulen muß, wird in DATA-Statements abgelegt. Die Ermittlung dieser Zeiten ist etwas mühselig; als Anhaltspunkt kann jedoch eine durchschnittliche Zeit von 1,2 s/KByte dienen. Mit etwas Geduld läßt sich diese Zeit dann bis zur gewünschten Genauigkeit optimieren.

Doch nun zum eigentlichen Programm. Der POKE-Befehl in Zeile 100 verhindert, daß der Recordermotor (auch bei gedrückten Tasten) Strom erhält. Dieser Befehl sollte nur im Programm angewendet und auch wieder von diesem aufgehoben werden (Zeile 160), da sonst ein RESET bzw. Ausschalten des Rechners nötig wird. Wenn am Recorder eine Taste gedrückt ist, so enthält Zelle 59412 den Wert $233_{10} \triangleq 11101001_2$, sonst $249_{10} \triangleq 11111001_2$. Diese beiden Werte unterscheiden sich nur im fünften Bit. Auf eben dieses Bit wird 59412 durch die WAIT-Befehle getestet. TI$="000000" setzt die „Uhr" des Rechners auf Null, und der anschließende POKE-Befehl schaltet den Recorder-Motor ein. Die POKE-Befehle in Zeile 170 schreiben direkt in den Tastatur-Puffer (dreimal Cursor nach oben, RETURN). POKE525,4 gibt an, daß vier gültige Zeichen im Tastaturpuffer stehen. Diese Befehlsfolge bewirkt, daß das auf dem Bildschirm stehende „LOAD" mit RETURN übernommen wird.

Dieter Smode

```
100 READ T:POKE59411,62
110 PRINT"STOP-TASTE DRUECKEN!":WAIT59412,16,233:PRINT"OK"
120 PRINT"SCHNELLEN VORLAUF EINSCHALTEN!":WAIT59412,16,249
130 PRINT"BITTE WARTEN.":TI$="000000":POKE59411,53
140 IF TI/60<=T THEN 140
150 POKE59411,62:PRINT"STOP-TASTE DRUECKEN!":WAIT59412,16,233
160 POKE59411,61:PRINT"LOAD"
170 POKE527,145:POKE528,145:POKE529,145:POKE530,13:POKE525,4:NEW
```

Auch wenn man mehrere Programme auf eine Kassette speichert, kann man sie ohne Bandzählwerk schnell wiederfinden: Das Menue-Programm benutzt den schnellen Vorlauf des Recorders

Mischprodukt-Errechnung in Basic

Das Errechnen von Mischprodukten mehrerer Oszillatoren und ihrer Harmonischen ist ein interessantes Problem der Kombinatorik. Betrachtet man nur diejenigen Mischprodukte, die nicht selbst durch Mischung von Mischprodukten zustande kamen, so lassen sich die zu erwartenden Ergebnisse wie folgt ausdrücken:

$$f = |a_1f_1 \pm a_2f_2 \pm a_3f_3 \pm \ldots a_nf_n|$$

f ist das entstehende Mischprodukt; die Faktoren $a_1 \ldots a_n$ sind die Harmonischen-Ordnungszahlen der Oszillatoren 1...n und können die Werte 0...h annehmen, wobei h die maximale Ordnungszahl ist, bis zu der gerechnet werden soll; $f_1 \ldots f_n$ sind die Oszillator-Grundfrequenzen.

Es ist leicht einzusehen, daß eine solche Berechnung sehr zeitaufwendig sein kann, besonders dann, wenn mehr als zwei Oszillatoren vorhanden sind. Aus diesem Grunde wurde ein Basic-Programm entwickelt (*Bild*), das diese Aufgabe automatisiert und – wenn man einmal von der Rechenzeit absieht – nahezu beliebig viele Oszillatoren und Harmonische zuläßt. Da außerdem meist nur Mischprodukte innerhalb eines bestimmten Frequenzbereiches interessieren, z. B. innerhalb des Empfangsbereiches eines Receivers, werden nur solche Werte ausgegeben, die zwischen eine untere und eine obere Frequenzgrenze fallen.

Das Programm ist ein wenig „trickreich" aufgebaut, weil ja eine beliebige Summandenzahl in obiger Gleichung auftreten kann. Es benutzt für die Faktoren a die Matrixvariable A(Z), wobei – um den Vorzeichenwechsel einzusparen – die Werte A(1) bis A(EW) den Bereich –h bis +h durchlaufen; nur A(0) ist immer Null bzw. positiv.

Das Programm im Bild wurde mit einem AIM-65 entwickelt und läuft ohne Änderung auch auf dem PET-2001 bzw. CBM-3001.

Herwig Feichtinger

```
 10 PRINT"§QQ   *** MISCHPRODUKTE ***":INPUT"QQQUNTERE GRENZE";U
 20 INPUT"OBERE GRENZE";O
 30 INPUT"OSZILLATOR-ANZAHL";VZ
 40 VZ=VZ-1:REM WEIL AUCH 0-INDEX MOEGL.
 50 INPUT"MAXIMALE HARMONISCHEN-ORDNG.";EW
 60 DIMA(VZ):DIMF(VZ)
 70 FORI=0TOVZ:A(I)=-EW:PRINT"F";I+1;:INPUTF(I):NEXT
 75 A(VZ)=0
 80 Z=0:REM INDEX
 90 S=0:REM QUERSUMME
100 FORI=0TOVZ:S=F(I)*A(I)+S:NEXT
110 S=ABS(S):IFS>UANDS<OTHENPRINTS,
120 A(Z)=A(Z)+1:IFA(Z)>EWGOTO140
130 GOTO90
140 A(Z)=-EW
150 Z=Z+1:IFZ>VZTHENPRINT"FERTIG":END
160 A(Z)=A(Z)+1
170 IFA(Z)>EWGOTO140
180 Z=0:GOTO90
READY.
```

Dieses Programm läuft ohne Änderungen auf den Computern AIM-65, PC-100, PET-2001, CBM-3001 und errechnet die Mischprodukte, die sich aus mehreren Oszillatorfrequenzen und deren Harmonischen ergeben

Statistikfunktionen des TI-58/59

Sicherlich werden sich viele Besitzer der programmierbaren Taschenrechner TI 58/59 schon gefragt haben, wie die Statistikfunktionen (OP 11 bis OP 15, $\bar{x}$ usw.) ablaufen. Es gibt eine Reihe von Möglichkeiten, einem TI dabei „auf die Finger" zu sehen. Beispielsweise bewirkt die Tastenfolge in *Tabelle 1* ([1], V3N11p1) – Eingabe nur über Tasten, nicht über Programmspeicher – die Anzeige eines mit „HIR 08 ((RCL 06 – RCL 04" beginnenden Programms, welches offenbar in einem ROM im TI gespeichert ist. Man kann sich dieses Programm mit SST (und gegebenenfalls gedrückter TRACE-Taste) oder mit LIST ansehen bzw. ausdrucken. Andere Tasten bringen die Anzeige zum Verlöschen, so daß Aus- und Einschalten des Rechners erforderlich wird.

Beim Programmausdruck mit LIST werden nach dem Schritt 487 die Schritte ab 039 wiederholt. Die Schritte ab 488 sieht man sich dann zweckmäßigerweise mit SST an, indem man den Rechner aus- und einschaltet, und der Tastenfolge in Tabelle 1 ein GTO 479 voranstellt (für TI 58: 0 OP 17 GTO 479). Die Schritte 512 bis 575 wiederholen die Schritte 384 bis 447.

Die Programmschritte 000 bis 379 enthalten die Programme für die Statistikfunktionen und (INV) D.MS sowie (INV) P/R. Die Adressen sind in *Tabelle 2* wiedergegeben. Der in diesem Programmabschnitten häufig vorkommende Code „HIR xy" (HIR ≙ 82) wirkt bekanntlich (z. B. [2]) auf die Anzeige und den Zwischenspeicher (für unvollständige Operationen) Nr. y (y = 1, 2...8) für x = 0, 1, 3, 4, 5, 6–9 wie STO, RCL, SUM, PRD, ISUM bzw. IPRD. Daneben hat noch der Befehl HIR 20 in den Programmschritten 045/046 und 082/083 eine Wirkung, nach [1], V3N11p2 vermutlich ähnlich wie GTO IND.

Die Schritte 384 bis 511 wurden von B. Skillman ([1], V3N12p5) als 16 16ziffrige Konstanten identifiziert: ln 10, ln 2, ln 1.1 bis ln 1.000001, π/4, arctan 0.1 bis arctan 0.0001, π/2, π, 18/π. Dabei ist die Funktion der jeweils letzten 3 Ziffern unklar.

Will man die Statistikprogramme wie Modulprogramme in den Programmspeicher übernehmen, so muß man sie nach der vorher erhaltenen Auflistung über die Tastatur eingeben. Im Programmspeicher ist allerdings HIR 20 offenbar wirkungslos; außerdem ist auch die Genauigkeit in beiden Fällen manchmal unterschiedlich. Beispielsweise ergibt die Anwendung von D.MS INV D.MS auf 2,5614 (nach [1], V3N11p2) 2.561399999920, während die entsprechende Folge, im Programmspeicher durchgeführt, 2.561399999999 ergibt.

Tabelle 1:
Aufruf eines ROM-Programms

Tastenfolge	Anzeige
10 OP 17	159.99 *)
PGM 12	159.99
SBR 333	159.99 ?
R/S	0. 00 ?
P/R	0 ?
LRN	000 82
*) 4 OP 17 für TI 58	159.39

Tabelle 2:
ROM-Adressen des TI-58/59

000 – 046	OP 12
000 – 057	OP 15
000 – 046 / 058 – 066	OP 14
067 – 083	$\bar{x}$
067 – 106	OP 11
107 – 148	INV $\bar{x}$
149 – 191	OP 13
192 – 212 / 236 – 249	Σ+
213 – 249	INV Σ+
250 – 283	INV P/R
284 – 302	P/R
303 – 340	D.MS
341 – 379	INV D.MS

Mit der Kenntnis dieser einzelnen Programme kann man nun verfolgen, welche Werte dabei in den Zwischenspeichern abgelegt werden, und wie diese Programme genau ablaufen.

Dr. Harald Lindner
Michael Lücker

Literatur

[1] 52-Notes, published by R.C. Vanderburgh in Dayton, Ohio, U.S.A., (V = Volume, N = Number, p = page).

[2] Tips für TI-59-Benützer. FUNKSCHAU 1980, Heft 5, Seite 80.

Der Software-Klau geht um

Zuweilen werden der FUNKSCHAU-Redaktion Programme zum Abdruck angeboten – gute und weniger gute, nützliche und unnütze. Daß wir hier eine Auswahl treffen müssen (und zum Glück auch können), ist wohl selbstverständlich.

Nun gibt es einige schlaue Zeitgenossen, die das, was man zuweilen bei Schaltungen fand und findet, auch auf Software anwenden. Sie nehmen nämlich ein irgendwo einmal veröffentlichtes Programm, ändern eventuell ein paar Kleinigkeiten, ohne aber die Gesamtstruktur zu ändern. Und das bieten sie dann einer Fachzeitschrift an.

Zugegeben, eine Redaktion ist nicht immer in der Lage, die gesamte bisherige Literatur durchzuwühlen und somit festzustellen, ob ein eingesandtes Programm nur eine Kopie darstellt. Es liegt also auch ein wenig am Autor, ob er sich das Armutszeugnis ausstellen will, einer Zeitschrift ein „geklautes" Programm anzubieten. Wenn es die Redaktion oder später die Leser merken, wird ihm womöglich das Honorar für die Veröffentlichung gestrichen.

Wenn man Programme schreibt, läßt es sich nicht immer umgehen, daß ein gewisser Teil daraus schon früher einmal irgendwo veröffentlicht war. Dann bricht sich der Autor bestimmt keinen Stein aus der Krone, wenn er seine verwendeten Quellen in einer Literaturangabe nennt.

Eine andere, oft zu findende Unsitte ist es, mehreren Zeitschriften gleichzeitig Manuskripte, Schaltungen oder Programme anzubieten. Hier sei gesagt: Die FUNKSCHAU veröffentlicht nur exklusive Beiträge, die sonst noch nirgendwo erschienen sind und die auch keiner anderen Zeitschrift angeboten sind. Noch ein Wort zu eingesandten Programmen: Bitte fügen Sie stets ein möglichst kontrastreiches, sauberes Listing bei, wobei es möglich sein sollte, eine Fotokopie zu erstellen, ohne daß einzelne Zeichen „ausfransen".

Übrigens: Mikrocomputer-Beiträge in der FUNKSCHAU werden bei Veröffentlichung mit 200 DM pro Druckseite honoriert. Ist das nichts?

Berichtigung

PET als Logikanalysator

FUNKSCHAU-Sonderheft „Mikrocomputer-Anwendungen", Seite 20

Beim Ausdruck des Programmlistings entstanden zwei Fehler, die zu Funktionsschwierigkeiten führen können. In Zeile 311 muß es nach dem zweiten Doppelpunkt richtig heißen:
IF L=1 THEN W=255-W

In Zeile 418 muß die erste geöffnete Klammer nach IF durch PEEK ersetzt werden:
418 IF PEEK (515)...

Um das Programm auf dem CBM-Betriebssystem laufen lassen zu können, muß man statt PEEK (515) jeweils PEEK (151) schreiben.

Michael Konz

AIM-65-Basic-Erweiterung

Das Basic des AIM-65 ist eine etwas abgemagerte Version des PET/CBM-Basic. Vor allem vermißt man Editiermöglichkeiten und die Möglichkeit, Daten auf Kassette speichern zu können. Dieses und vieles andere gestattet die hier beschriebene 2-KByte-Erweiterung.

Die Erweiterung liegt im Bereich ab 0220 (*Bild*), so daß das im Arbeitsspeicher abgelegte Basic-Programm nach oben verschoben wird. Das hat gegenüber einer Plazierung der Erweiterung am oberen RAM-Ende den Vorteil, daß bei einer RAM-Erweiterung ab 1000 sich automatisch der Interpreter mehr Arbeitsspeicher reserviert, ohne daß man das Erweiterungsprogramm auf einen anderen Adressenbereich umschreiben muß. Die Initialisierung erfolgt zunächst wie gewohnt mit Taste 5. Dann lädt und startet man die Erweiterung bei 0220 und ist sofort in Basic (Warmstart). Jetzt stehen 19 zusätzliche Befehle zur Verfügung.

Die Befehle

AUT# Zeilennummer, Inkrement

schaltet die automatische Zeilennumerierung ein. Nach jedem Drücken von RETURN wird die Zeilennummer um einen vorher bestimmten Wert erhöht. Das Eingeben durch den Programmierer entfällt dadurch. Der erste Parameter nach AUT# bedeutet die Zeilennummer, ab der begonnen werden soll, der zweite die Schrittweite.

Beispiel:

AUT# 100,5 bewirkt eine Numerierung ab Zeile 100 in Fünferschritten (100, 105, 110,...).

Wird der Befehl ohne weitere Angaben gegeben, dann nimmt das Programm für beide Parameter den Wert 10 an.

Dieser Betriebsmodus kann nur durch CTRL Z wieder verlassen werden. Dabei wird die sich im Buffer befindende Zeile gelöscht, ohne vorher in den Hauptspeicher übernommen zu werden. AUT# ist nicht programmierbar. Erscheint der Befehl mit einer Zeilennummer versehen in einem Programm, so wird dieses mit einem ?ID ERROR abgebrochen. Das gleiche gilt auch für die beiden folgenden Befehle FETCH und HELP.

FETCH Zeilennummer

holt die angegebene Zeile in den Eingabepuffer, wo sie editiert werden kann. Existiert keine Zeile mit der spezifizierten Nummer, folgt ein ?US ERROR.

Mit diesem Befehl ist es möglich, Programmzeilen zu ändern, ohne sie vollständig neu eingeben zu müssen. Die ersten neunzehn Zeichen der Zeile werden zusammen mit einem Cursor (_) im Display dargestellt. Der Cursor wird über folgende Tasten gesteuert:

SHIFT F1 ... um eine Stelle nach links

SHIFT DEL... um eine Stelle nach rechts

DEL... um eine Stelle nach links, das links vom Cursor stehende Zeichen wird gelöscht

SHIFT F2 ... Rückspeichern der Zeile im Puffer in den Hauptspeicher und Übernahme der Zeile mit der nächst größeren Nummer in den Puffer (Cursor down)

SHIFT F3 ... Cursor up (wie SHIFT F2)

Wo der Cursor sich gerade befindet, können Zeichen eingefügt werden. Der rechts stehende Text wandert dabei automatisch weiter, bis der Puffer, der maximal 71 Zeichen faßt, gefüllt ist.

Der Editiermodus wird mit RETURN verlassen. Dabei wird die Zeile wieder in den Hauptspeicher übernommen. Beim Verlassen des Modus mit ESC geschieht das nicht, man landet aber im Monitor.

HELP

dient zur Lokalisierung von Fehlern in einer Programmzeile. Wird der Befehl unmittelbar nach einer Fehlermeldung, die während einer Programmausführung auftrat, gegeben, so wird die fehlerbehaftete Zeile angezeigt und die Stelle, die den Fehler verursacht hat, mit einem Fragezeichen markiert. Der Befehl ist unwirksam, wenn der Fehler im Direktmodus verursacht wurde oder zwischenzeitlich andere Anweisungen ausgeführt worden sind.

OPEN "Filename",m

öffnet ein sequentielles File auf Kassette. Es werden nur die ersten fünf Zeichen des Namens anerkannt. Der Name muß in Anführungszeichen stehen, d. h. eine Stringvariable ist hier nicht erlaubt.

Von dem Parameter m hängt es ab, ob Daten geschrieben oder gelesen werden sollen. Für eine Leseoperation besitzt m den Wert 0, für eine Schreiboperation den Wert 1. Soll eine Leseoperation ausgeführt werden, erscheint sofort die Meldung ‚READING TAPE' auf dem Display. Die Programmausführung wird unterbrochen, bis der erste Block des Files geladen ist. Ein Abbruch ist an dieser Stelle nur mit RESET möglich.

Analog hierzu erscheint die Meldung „WRITING TAPE", wenn der Ausgabebuffer gefüllt ist und ein Block von 80 Byte auf Band geschrieben wird. Sinnvollerweise sollte der Computer die Möglichkeit haben, den Motor des Kassettenrecorders zu steuern. Besonders beim Lesen vom Band ist das wichtig, sonst kann es geschehen, daß das Programm ewig auf einen Datenblock wartet, der längst vorüber ist.

CLOSE

schließt ein zuvor geöffnetes File wieder. Wurde zuvor eine Leseoperation ausgeführt, dann kann CLOSE entfallen. Nach einer Schreiboperation ist es jedoch obligatorisch, da sonst der letzte Datenblock nicht übertragen wird.

WRITE

Dieser Befehl wird in der gleichen Syntax wie PRINT benutzt. Die Daten werden aber auf Kassette geschrieben. Ein File muß dort zuvor mit OPEN geöffnet worden sein.

TAPIN

In Analogie zu INPUT liest TAPIN Daten von einem externen File ein. Zwar können nach TAPIN ebenfalls Promptingtexte in Anführungszeichen folgen, wie nach INPUT, sie werden bei der Ausführung des Befehles aber unterdrückt. Im Zusammenhang mit den letzten vier Befehlen können verschiedene Fehlermeldungen auftreten:

?FC ERROR

a) Ein bereits geöffnetes File soll erneut geöffnet werden.

b) Ein WRITE oder TAPIN ohne vorangegangenes OPEN wurde entdeckt.

c) Ein- und Ausgabe sind gleichzeitig auf das Kassetteninterface geschaltet. Das Betriebssystem würde dann automatisch den in der Zero-Page gelegenen zweiten Kassettenbuffer mitbenutzen und dadurch Teile der Interpreter-Variablen überschreiben.

?OM ERROR

FOR-NEXT-Schleifen oder Unterprogramme sind so tief verschachtelt, daß der Kassettenbuffer, der ja in der Page 1 liegt, diese Daten zerstören würde.

Zwei Beipiele sollen die Verwendung dieser vier Befehle verdeutlichen:

```
10 OPEN „TEST“,1
20 FOR I= 1 TO 50
30 WRITE I
40 NEXT
50 CLOSE
```

Das zweite Beispiel:

```
10 OPEN „TEST“,0
20 FOR I= 1 TO 50
30 TAPIN X
40 PRINT X
50 NEXT
60 CLOSE
```

Das erste Programmbeispiel schreibt die Zahlen von 1 bis 50 auf Band, das zweite Programm liest diese Zahlen wieder ein und druckt sie aus.

TIME

Mit dem Befehl TIME kann die Softwareuhr abgerufen werden, die zusammen mit der Basic-Erweiterung zur Verfügung steht. TIME kann wie eine gewöhnliche Variable angesprochen werden. Ihr Wert ist aber vom Programm her nicht änderbar. Erscheint

```
[M]=0220 A2 03 BD ED 09 9D 08 01 CA 10 F7 A9 9C A0 02 85
[ ] 0230 01 84 02 A9 4C 85 BF A9 B8 A0 02 8D 00 A4 8C 01
[ ] 0240 A4 A9 40 8D 0B A0 A9 C3 A0 50 8C 04 A0 8D 05 A0
[ ] 0250 A9 C0 8D 0E A0 A2 14 A9 00 95 EB CA 10 FB A9 A0
[ ] 0260 85 EF 24 C2 30 10 A5 7F 38 66 C2 E9 0F 85 7F B0
[ ] 0270 02 C6 80 20 1F 0A 20 8D 02 A0 10 A9 00 91 7F 88
[ ] 0280 10 FB A9 E0 A0 02 85 C3 84 C4 4C 7F B2 A9 C6 A0
[ ] 0290 02 D0 04 A9 E2 A0 06 85 C0 84 C1 60 A5 C6 A4 C7
[ ] 02A0 85 A2 84 A3 A5 82 85 A9 AD 12 A4 C9 54 F0 06 A9
[ ] 02B0 55 8D 12 A4 58 4C 00 B9 2C 04 A0 E6 F3 D0 06 E6
[ ] 02C0 F2 D0 02 E6 F1 40 86 FF BA BD 01 01 C9 0B D0 10
[ ] 02D0 BD 02 01 C9 BE D0 09 A0 01 B1 C6 20 6E BE B0 0B
[ ] 02E0 A6 FF E6 C6 D0 02 E6 C7 4C C5 00 A5 C6 D0 02 C6
[ ] 02F0 C7 C6 C6 A0 FF 84 FE A2 FF C8 E8 BD 7A 09 F0 3C
[ ] 0300 38 F1 C6 F0 F4 C9 80 F0 0C A0 00 E6 FE E8 BD 79
[ ] 0310 09 10 FA 30 E6 20 6A B7 E6 FE 06 FE A6 FE E0 03
[ ] 0320 B0 07 A5 C7 F0 03 4C E9 C0 E0 10 F0 17 68 68 68
[ ] 0330 68 BD C7 09 48 BD C8 09 48 4C BF 00 A0 01 20 6A
[ ] 0340 B7 6C C3 00 A9 EF A0 00 20 E1 C8 06 AA 20 16 C6
[ ] 0350 20 06 C9 86 8F 84 90 68 68 20 BF 00 8A 60 B0 20
[ ] 0360 20 DA B7 F0 1A A5 14 A4 15 85 FA 84 FB 20 C5 BC
[ ] 0370 20 DA B7 A5 14 A4 15 85 FC 84 FD 38 66 F7 60 18
[ ] 0380 A9 0A A0 00 85 FA 84 FB B0 ED 90 E7 20 6B BB 20
[ ] 0390 36 C5 6C 14 00 A6 F5 D0 56 C9 22 D0 55 20 85 BC
[ ] 03A0 A4 A9 C0 05 90 02 A0 04 B1 AA 99 2E A4 88 10 F8
[ ] 03B0 20 BF 00 38 66 F5 48 20 BF 00 68 A2 00 C9 30 F0
[ ] 03C0 0E 8E 35 A4 20 DC 03 A9 40 8D 09 A4 4C 6F E5 8E
[ ] 03D0 34 A4 A2 0D 20 DC 03 46 F5 4C 2F E3 20 EA 03 BD
[ ] 03E0 05 0A F0 2A 20 05 EF E8 D0 F5 A9 0D 4C 05 EF 4C
[ ] 03F0 87 BF 4C D0 BC 20 1A 04 24 F5 10 F3 A2 54 EC 12
[ ] 0400 A4 F0 EC 20 11 04 20 BA B8 A5 EE 8D 13 A4 60 A2
[ ] 0410 55 AD 13 A4 85 EE 8E 13 A4 60 BA E0 70 B0 EF 4C
[ ] 0420 57 B2 24 F5 10 03 20 11 E5 20 00 00 A9 00 85 F5
[ ] 0430 A9 08 8D 09 A4 60 24 F5 50 B5 20 1A 04 A2 54 EC
[ ] 0440 13 A4 F0 AB AD 12 A4 48 8E 12 A4 38 66 10 20 BE
[ ] 0450 B9 06 10 68 8D 12 A4 60 B0 98 20 DA B7 20 36 B4
[ ] 0460 B0 03 4C 52 B7 66 F4 20 0F 04 A2 FF 86 FF 20 44
[ ] 0470 05 20 09 04 A6 FF 86 ED A0 04 84 FE A4 FE A6 ED
[ ] 0480 B1 A7 95 16 F0 0F 10 07 38 E9 7F 20 AD 05 2C E6
[ ] 0490 ED E6 FE D0 E7 E0 13 90 02 A2 12 86 ED 20 11 05
[ ] 04A0 20 88 05 30 F8 C9 7F D0 12 A6 ED F0 F3 CA 86 ED
[ ] 04B0 B5 17 95 16 E8 E0 47 90 F7 B0 E2 C9 0D F0 09 20
[ ] 04C0 FB 04 B0 DC 95 16 90 D5 46 F4 48 48 A9 7E A0 B2
[ ] 04D0 D0 04 A9 97 A0 06 8D FE 01 8C FF 01 20 EA 03 38
[ ] 04E0 66 EC A2 15 A0 00 4C 87 B2 68 86 FE A6 FF 95 16
[ ] 04F0 E6 FF A6 FE 24 F4 30 18 4C 7A E9 48 A2 46 E4 ED
[ ] 0500 F0 0D CA B5 16 95 17 E4 ED D0 F7 E6 ED E0 46 68
[ ] 0510 60 A2 00 8E 15 A4 E4 ED D0 06 A9 DF 20 05 EF 2C
[ ] 0520 B0 1D B5 17 08 B5 16 28 F0 08 09 80 20 05 EF E8
[ ] 0530 D0 E4 E8 E4 ED D0 05 20 05 EF A9 5F 4C 05 EF E0
[ ] 0540 13 90 DF 60 A0 02 B1 A7 AA C8 B1 A7 4C 0C CB 86
[ ] 0550 FF 24 EC 10 06 46 EC A2 FD 9A 60 24 F7 10 1D 8A
[ ] 0560 D0 1A 20 0F 04 A5 FB A6 FA 20 0C CB 20 09 04 18
[ ] 0570 A5 FA 65 FC 85 FA A5 FB 65 FD 85 FB AE 15 A4 E0
[ ] 0580 14 B0 05 A9 DE 20 7B EF 20 40 EC E0 10 F0 65 E0
[ ] 0590 20 B0 0C C9 1B D0 05 46 F4 4C 56 E9 A6 FF 60 C9
[ ] 05A0 41 90 F9 C9 5B B0 6E E0 40 F0 3D E9 11 A2 90 A0
[ ] 05B0 B0 86 A0 84 A1 AA A0 FF CA F0 07 C8 B1 A0 10 FB
[ ] 05C0 30 F6 A6 FF 24 F4 10 05 20 FB 04 86 FF C8 B1 A0
[ ] 05D0 48 29 7F E0 47 B0 04 95 16 E6 FF 24 F4 30 03 20
[ ] 05E0 6C B9 A6 FF 68 10 DD 60 C9 54 B0 B0 E9 3F A2 7A
[ ] 05F0 A0 09 D0 BD C9 2C 90 04 E9 2B B0 B1 C9 15 B0 04
```

Bei der Eingabe des 2-K-Maschinenprogramms sollte man tunlichst das im gleichen Heft abgedruckte Prüfsummen-Programm benutzen, um Eingabefehler zu vermeiden. Die Basic-Erweiterung bietet bei den Mikrocomputern AIM-65 und PC-100 neunzehn äußerst nützliche neue Basic-Befehle

TIME auf der linken Seite einer Wertzuweisung, dann bleibt diese Zuweisung wirkungslos.

TIME liefert die Zeit in $^1/_{20}$ Sekunden und benutzt den Timer 1 im User-VIA. Die Uhr wird durch einen RESET angehalten und muß dann neu initialisiert werden, wie dies zu Beginn beschrieben worden ist. Das folgende Programmbeispiel bildet auf dem Display eine Digitaluhr nach:

```
10 T  = ROUND(TIME/20,0)
20 ST = INT(T/3600): T=T-3600*ST
30 MI = INT(T/60): T=T-60*MI
40 IF T = T1 THEN 10
50 T1 = T
60 PRINT ST;":";MI;":";T
70 GOTO 10
```

Der Befehl ROUND in Zeile 10 wird später noch erläutert werden.

```
[ ] o6oo 69 1o 9o A9 C9 1A Do 53 2o oo B9 A2 oo 86 FF 86
[ ] o61o F4 86 F7 Fo oE 24 F4 1o 46 C9 5B Do o9 A6 ED Fo
[ ] o62o o2 C6 ED A9 FF 6o C9 7F Do oF A6 ED E8 Eo 47 Bo
[ ] o63o F2 B5 15 Fo EE 86 ED Do EA C9 5D Do 33 Ao o1 B1
[ ] o64o A7 85 F9 88 B1 A7 85 F8 C8 B1 F8 Fo D6 C8 B1 F8
[ ] o65o AA C8 B1 F8 85 F9 86 F8 4C D2 o4 C9 18 Do o5 A9
[ ] o66o oC A6 FF 6o 68 68 68 68 68 68 2o 87 B6 4C C8 B5
[ ] o67o A5 73 A4 74 85 F8 84 F9 Ao o1 B1 F8 AA C5 A8 9o
[ ] o68o oE Do Ao 88 B1 F8 C5 A7 9o o8 Do 97 C8 Do BE 88
[ ] o69o B1 F8 86 F9 85 F8 9o Eo A5 F8 A4 F9 85 14 84 15
[ ] o6Ao 4C 5D o4 Fo 2C Ao oo 84 F8 2o E4 BD A5 8D A4 F8
[ ] o6Bo 91 7F C8 A5 8E 91 7F C8 Co 1o 9o o3 4C 57 B2 84
[ ] o6Co F8 2o C5 oo Fo o8 2o C5 BC Do DE 4C Do BC 91 7F
[ ] o6Do C8 66 F6 4C 93 o2 46 F6 46 EB 4C 8D o2 38 66 EB
[ ] o6Eo 3o F1 86 FF BA BD o1 o1 C9 FA Do o7 BD o2 o1 C9
[ ] o6Fo B5 Fo o7 24 F6 3o 3E 4C C8 o2 A5 C7 Fo F9 A5 81
[ ] o7oo A6 82 C5 83 Do o4 E4 84 Fo ED 85 83 86 84 98 48
[ ] o71o 24 EB 1o o7 A9 oo Ao oA 2o 4A B9 2o o1 CB 2o oo
[ ] o72o B9 68 24 EB 1o D1 49 FF 38 65 C6 Bo o2 C6 C7 85
[ ] o73o C6 18 4C 61 B6 BD o1 o1 C9 1F Do BB BD o2 o1 C9
[ ] o74o B8 Do B4 Ao FE C8 C8 Co 1o Bo AC B1 7F Fo A8 C5
[ ] o75o 8D Do F2 C8 B1 7F C5 8E Do EC A5 8D 29 7F 2o 6C
[ ] o76o B9 A5 8E 29 7F Fo o3 2o 6C B9 A9 25 24 oB 3o o6
[ ] o77o 24 oA 1o o5 A9 24 2o 6C B9 A9 3A 2o 6C B9 A5 91
[ ] o78o A4 92 2o DF BC A5 oB 48 46 oB A5 oA 48 2o AE o7
[ ] o79o 2o 67 B9 68 85 oA 68 85 oB 2o BF oo 2o 2o B8 68
[ ] o7Ao 68 A9 2F 2o 6C B9 2o AE o7 E6 oA 4C oo B9 24 oA
[ ] o7Bo 3o 11 24 oB 1o o7 A5 AC A4 AD 2o D1 Co 2o 1C CB
[ ] o7Co 2o C5 C1 4C 4D B9 A5 81 A4 A9 85 14 84 15 2o 36
[ ] o7Do B4 9o 36 A5 A2 E5 A7 85 F8 A5 A3 E5 A8 Do 2A 2o
[ ] o7Eo 44 o5 Ao o3 84 FE A2 oo 86 FF A4 FE C8 84 FE C4
[ ] o7Fo F8 Do o3 2o 6A B9 B1 A7 Fo oF 1o o8 38 E9 7F 2o
[ ] o8oo AD o5 3o E2 2o 6C B9 Do DD 68 68 4C 7F B2 2o B9
[ ] o81o BC 2o 7o BB Ao oo B1 AC 85 FE 84 FF 84 A9 C8 B1
[ ] o82o AC 85 F8 C8 B1 AC 85 F9 A5 AC A4 AD 2o 19 C4 A4
[ ] o83o FF C4 FE Fo 1A B1 F8 2o 7D EA 9o o3 4C 87 BF 48
[ ] o84o A9 F1 Ao o9 2o 67 C7 68 2o BD CA E6 FF Do Eo A9
[ ] o85o oo 85 oA 6o 2o B9 BC 2o 6E BB 4C 5A C9 2o 54 o8
[ ] o86o A5 A9 Fo EF 38 E9 8o 9o 1E A8 Co o8 9o 25 A5 AB
[ ] o87o 85 AA A5 AC 85 AB A5 AD 85 AC A9 oo 85 AD 98 E9
[ ] o88o o8 Do E6 A9 8o 85 A9 4C 16 C6 o6 AD 26 AC 26 AB
[ ] o89o 26 AA 88 98 Do F4 Fo EB 2o 54 o8 46 AE 2o oB CA
[ ] o8Ao A5 A9 Fo 5o C9 C3 Bo 55 A9 oo 85 FF 2o 4B C9 38
[ ] o8Bo E9 o4 85 A9 2o 62 o8 49 FF 38 69 84 A8 A5 AA 4A
[ ] o8Co 88 Do FC 69 3o C9 3A 9o o2 69 o6 48 E6 FF 2o 3B
[ ] o8Do C9 38 E9 o4 85 A9 2o oB CA A8 Do Do A2 oo A4 FF
[ ] o8Eo 68 9D FF o1 E8 88 Do F8 A9 oo 9D FF o1 A9 FF Ao
[ ] o8Fo o1 4C C5 C1 A9 3o 8D FF o1 A2 o1 Do EB A2 1A 4C
[ ] o9oo 59 B2 2o C1 BC 2o 6B BB A5 AE 48 46 AE 2o o9 C9
[ ] o91o 2o C5 BC 2o DC C4 2o BF BC Eo oA 9o o3 4C 87 BF
[ ] o92o 8A Fo 25 86 FE 86 FF A9 9F Ao oo 2o E1 C8 2o 21
[ ] o93o C8 C6 FE Do F9 2o 88 C5 2o oB CA 2o 3D C8 C6 FF
[ ] o94o Do F9 68 85 AE 4C 5A C9 A9 9F Ao oo 2o E1 C8 2o
[ ] o95o 88 C5 4C 5D o9 2o 54 o8 A5 AE 48 46 AE 2o oB CA
[ ] o96o 68 85 AE 6o 2o 54 o8 A9 F6 Ao o9 2o 67 C7 4C 5A
[ ] o97o C9 2o 54 o8 A9 FB Ao o9 Do F1 41 55 54 A3 46 45
[ ] o98o 54 43 C8 48 45 4C Do 44 45 C3 44 45 C7 46 52 41
[ ] o99o C3 54 41 5o 49 CE 48 45 58 A4 54 49 4D C5 53 59
[ ] o9Ao D3 54 52 41 43 C5 43 4C 4F 53 C5 52 41 C4 54 52
[ ] o9Bo 4E C3 4F 5o 45 CE 57 52 49 54 C5 4F 46 C6 52 4F
[ ] o9Co 55 4E C4 53 53 D4 oo o3 5D o4 57 o7 C5 o8 oD o9
[ ] o9Do 7o o8 5C o4 35 o8 97 o3 43 o3 8B o6 A2 o4 21 o9
[ ] o9Eo 63 o9 54 o3 94 o3 F4 o6 D5 o9 o1 o6 DC 4F o5 E9
[ ] o9Fo o4 85 oo oo oo oo 7B oE FA 35 12 86 65 2E Eo B3
[ ] oAoo 53 54 45 5o oo 57 52 49 54 49 4E 47 2o 54 41 5o
[ ] oA1o 45 oo 52 45 41 44 49 4E 47 2o 54 41 5o 45 oo A9
[ ] oA2o 2B 85 73 A9 oA 85 74 4C 67 B4 oo oo oo oo oo oo
```

SYS numerischer Ausdruck

ruft ein Maschinenprogramm auf, das an der angegebenen Adresse (dezimal), beginnt. Das Maschinenprogramm muß mit RTS abgeschlossen sein.

TRACE

ist einer von drei Befehlen, die das Debugging von Programmen erheblich erleichtern. Bei jedem Übergang in eine neue Programmzeile wird die Meldung „IN n" ausgegeben, wobei n die betreffende Zeilennummer ist. Zusätzlich bietet TRACE die Möglichkeit, bis zu acht Variablen zu spezifizieren, die dann verfolgt werden können.

Das bedeutet, daß bei jeder Wertänderung einer der Variablen ihr alter und neuer Wert zusammen mit dem Namen und der Zeilennummer ausgedruckt werden. Die Grundfunktion von TRACE bleibt dabei unberührt.

Vorsicht ist bei Array-Variablen geboten. Der TRACE-Befehl unterscheidet nämlich die einzelnen Elemente von Arrays nicht. So kann die Meldung A: 15.2 / 10 beispielsweise bedeuten, daß die Variable A den Wert 10 erhalten hat, aber die Zuweisung kann auch an irgendein Element des Arrays A erfolgt sein.

Ferner wird für die Laufvariable einer FOR-NEXT-Schleife nur die erste Wertzuweisung angezeigt, alle weiteren nicht. Aber selbst mit diesen Einschränkungen vereinfacht TRACE die Fehlersuche in umfangreichen Programmen wesentlich.

```
10 TRACE A,B$,C%
20 A=5
30 B$="TEST"
40 C%=200
50 A=18.7
60 B$="NEUER WERT"
70 C%=1
```

Obiges Programmbeispiel liefert folgenden Ausdruck, sofern der Drucker eingeschaltet ist:

```
A: 0 / 5
IN 30
B$: /TEST
IN 40
C%: 0 / 200
usw.
```

SST

gehört in die gleiche Kategorie wie TRACE. Mit der Ausführung von SST schaltet das Programm in einen Einzelschrittmodus, d. h. es wird immer nur eine Programmzeile ausgeführt. Danach erfolgt Programmabbruch mit der Meldung ‚STEP IN n'. Hierbei bedeutet n die gerade begonnene, aber noch nicht ausgeführte Zeile. Es können nun Variablen abgefragt und geändert werden. Das Programm kann durch den bekannten CONT-Befehl oder einfach durch CTRL W fortgesetzt werden.

OFF

hebt sowohl TRACE als auch SST wieder auf, ohne jedoch die Variablenliste des TRACE zu löschen. Wird erneut ein TRACE-Befehl erteilt, dann wird wieder die alte Variablenliste benutzt und braucht nicht neu eingegeben zu werden.

DEC (string)

ist einer von mehreren Umwandlungsbefehlen. DEC liefert das dezimale Äquivalent zu einer als beliebiger Stringausdruck im Argument stehenden hexadezimalen Zahl.

```
10 INPUT A$
20 A=DEC (A$)
30 PRINT A
40 GOTO 10
```

Das Kurzprogramm kann zur schnellen Umwandlung von Hexadezimalzahlen in ihre dezimale Darstellung verwendet werden. Enthält der Stringausdruck im Argument von DEC ein nicht-hexadezimales Zeichen, so erfolgt ein ?FC ERROR.

HEX$ (numerischer Ausdruck)

ist der Gegenbefehl zu DEC. Die Funktion liefert einen String, der das Argument hexadezimal wiedergibt.

Ein Argument, das größer als 7.3 x 10^{19} ist, führt zu einem ?LS ERROR, da der erzeugte String nicht in den Ausgabebuffer des Interpreters paßt, wo er zwischengespeichert wird.

Nur der ganzzahlige, positive Teil des Argumentes wird verwertet. –7.3 würde von der Funktion also beispielsweise als 7 interpretiert werden.

FRAC (numerischer Ausdruck)

trennt die Vorkommastellen des Argumentes ab und verwertet nur den Nachkommateil:
FRAC(–12.34) liefert –.34.

TRNC (numerischer Ausdruck)

ist der Gegenbefehl zu FRAC. Auf den ersten Blick wirkt TRNC wie die bekannte INT-Funktion. Für positive Zahlen trifft das auch zu. Man erinnere sich aber an die Definition der Integer-Funktion. Sie liefert die „größte ganze Zahl, die kleiner oder gleich dem Argument ist". Für –12.34 wäre das die Zahl –13. TRNC liefert hier als Resultat aber –12, d. h. es werden wirklich die Vorkommastellen ausgewertet.

ROUND (numerischer Ausdruck, d)

erleichtert die Rundung von Zahlen auf d Nachkommastellen. Während d ganzzahlige Werte zwischen 0 und 9 annehmen darf, ist der erste numerische Ausdruck beliebig.

```
10 FOR I=0 TO 9
20 PRINT ROUND (1.23456789,I)
30 NEXT
```

Das Beispiel liefert folgende Ausgabe:

```
1
1.2
1.23
1.235
1.2346
1.23457
1.234568
1.2345679
1.23456789
1.23456789
```

Die Rundung, die in der letzten Stelle auftritt, ist deutlich zu erkennen.

Tabelle 1: „Short-hand"-Tastenbelegung

Grundfunktion	CTRL	SHIFT left	SHIFT right
,	END	<	<
–	FOR	=	=
.	NEXT	>	>
/	DATA	?	?
0	INPUT	0	0
1	DIM	!	!
2	READ	"	"
3	LET	#	#
4	GOTO	$	$
5	RUN	%	%
6	IF	&	&
7	RESTORE	'	'
8	GOSUB	(	(
9	RETURN	)	)
:	REM	*	*
;	STOP	+	+
A	ON	SGN	AUT#
B	NULL	INT	FETCH
C	WAIT	ABS	HELP
D	LOAD	USR	DEC
E	SAVE	FRE	DEG
F	DEF	POS	FRAC
G	POKE	SQR	TAPIN
H	PRINT	RND	HEX$
I	CONT	LOG	TIME
J	LIST	EXP	SYS
K	CLEAR	COS	TRACE
L	GET	SIN	CLOSE
M	NEW	TAN	RAD
N	TAB(	ATN	TRNC
O	TO	PEEK	OPEN
P	FN	LEN	WRITE
Q	SPC(	STR$	OFF
R	THEN	VAL	ROUND
S	NOT	ASC	SST
T	STEP	CHR$	T
U	[1]	LEFT$	U
V	[1]	RIGHT$	V
W	[1]	MID$	W
X	[2]	GO	X
Y	[1]	[3]	Y
Z	[4]	[3]	Z

Bemerkungen:
[1] CTRL U, V, W, Y bewirken im SST-Modus die Fortsetzung des Programmes
[2] CTRL X sendet ein Formfeed aus ($0C)
[3] Diese Kombinationen sollten nicht verwendet werden
[4] Löschen der Zeile und Ausschalten des Auto-Modus

DEG (numerischer Ausdruck)

bewirkt die Umwandlung eines in Bogenmaß gegebenen Winkels in Altgrad.

RAD (numerischer Ausdruck)

bewirkt entsprechend dem Befehl DEG die Umwandlung eines in Altgrad angegebenen Winkels in Bogenmaß.

Tastaturbelegung

Als zusätzlichen Service bietet die Basic-Erweiterung die Belegung einer jeden Taste des AIM-65 mit einem Basic-Befehl. *Tabelle 1* gibt Aufschluß darüber, wie diese Belegung erfolgte. Zumindest die wichtigsten dieser Tastenkombinationen dürften sich nach kurzer Zeit eingeprägt haben. Zudem bieten die Stirnseiten der Tasten ausreichend Platz für eine geeignete Beschriftung.

Wie funktioniert es?

Nachdem nun die einzelnen Befehle kurz vorgestellt worden sind, soll jetzt die prinzipielle Arbeitsweise der Erweiterung besprochen werden.

Wichtigster Umstand, dem die Erweiterung ihre Funktion verdankt, ist die Tatsache, daß bei der Initialisierung des Interpreters die Routine, die die Zeichen aus dem Basic-Programm ausliest, in den RAM-Bereich kopiert wird. Diese Routine findet man an den Adressen 00BF...00D6. Es wird hier mit absoluter modifizierter Adressierung [1] gearbeitet. Bei jedem Aufruf wird der Pointer C6/C7 um ein erhöht, bevor über ihn ein Zeichen aus dem Basic-Programm geholt wird.

An diese Stelle setzt die Erweiterung bei ihrer Initialsierung einen JMP-Befehl zu einer eigenen Decodierroutine

bei D0A6. Dort wird zunächst durch Überprüfung der auf dem Stack gespeicherten Rücksprungadresse festgestellt, woher das Programm kam. Wurde die Routine von BE09 aus angesprungen, so hat der Interpreter einen Variablennamen mit mehr als zwei Zeichen im Programm entdeckt. Für die Erweiterung bedeutet das aber, daß möglicherweise einer der neuen Befehle vorliegt. Wird der vermeintliche Variablenname nicht in der Tabelle der Schlüsselwörter gefunden, handelt es sich wirklich um einen Variablennamen und die Programmausführung wird normal fortgesetzt. Fällt die Abfrage dagegen positiv aus, verzweigt die Erweiterung und führt den neuen Befehl aus.

Zusätzlich wird bei der Initialisierung der Erweiterung die Eingabe auf U, d. h. ‚user defined', umgeschaltet. Die dazugehörige Routine findet man bei 054F. Dort wird auch ein Flag abgefragt, das für die automatische Zeilennumerierung zuständig ist.

Die Befehle TRACE und SST ändern nun ihrerseits wieder den von der Erweiterung zuvor gesetzten Vektor. Die Überprüfung, von wo das Programm gekommen ist, muß nun umfassender betrieben werden. Die Herkunft von B5F8 ist ein Zeichen dafür, daß auf eine neue Zeile des Basic-Textes gesprungen wurde. In Abhängigkeit von verschiedenen Flags wird jetzt das Programm abgebrochen (im Einzelschrittmodus), die Zeilennummer ausgegeben oder eine weitere Überprüfung vorgenommen, ob nämlich eine LET-Anweisung vorliegt.

Letzteres ist der Fall, wenn das Programm von B81D kommt. Der alte Variablenwert wird ausgegeben und das LET vollendet. Anschließend folgt der neue Wert.

Die bei den Umwandlungsfunktionen (DEC, HEX$) benutzten Algorithmen sind recht einfach. Die Darstellung von Dezimalzahlen in Microsoft-Basic sei hier als bekannt vorausgesetzt [2].

Während DEC fortlaufend zuerst mit 16 multipliziert und dann das neue Digit addiert dividiert HEX$ durch 16 und berücksichtigt jeweils das Ergebnis vor dem Komma. Dabei wird FRAC als Unterprogramm aufgerufen. Die erzeugten hexadezimalen Ziffern werden im Ausgabebuffer 0200...0210, der gewöhnlich zur Umrechnung von Zahlen in Floating-Point-Darstellung in ihre Dezimaldarstellung verwendet wird, gespeichert. Daher resultiert auch die Begrenzung auf den Wertebereich bis $7{,}3 \times 10^{19}$.

Aus der Funktionsweise der Erweiterung folgt notwendigerweise, daß die Arbeitsgeschwindigkeit des Interpreters geringfügig verlangsamt wird. Diese Verlangsamung hält sich indes in Grenzen, vermeidet man die Verwendung von Variablennamen mit mehr als zwei Zeichen, da sonst jedesmal die gesamte Schlüsselworttabelle der Erweiterung abgefragt werden muß, wenn auf eine solche Variable zugegriffen wird.

Es ist möglich, die Erweiterung wieder abzuschalten, um sie dann später ohne Datenverlust reaktivieren zu können. Dazu müssen einige Speicherplätze manuell geändert werden:

0001 := 00
0002 := B9
A00E := 40

Bei 00BF müssen drei Befehle eingefügt werden:

00BF INC C6
00C1 BNE 02
00C3 INC C7

Die Erweiterung belegt in der Zero-Page die Plätze EC...FF und zieht von dem für Basic-Programme zur Verfügung stehenden Speicherraum die obersten 16 Byte für die Variablenliste des TRACE-Befehls ab.

Zu beachten ist, daß nach einem LOAD die Belegung der Tasten mit den Befehlswörtern nicht mehr aktiviert ist. Um sie wieder herzustellen genügt die Anweisung POKE 42002,85. Um das Verständnis zu erleichtern, gibt *Tabelle 2* die Adressen einiger wichtiger Routinen des Interpreters und deren Bedeutung wieder.

Literatur

[1] *Joepgen, Hans-Georg:* „6502 simuliert neue Adressierungsart", FUNKSCHAU-Sonderheft Programme.
[2] *Handle, Franz:* „Zahlendarstellung im PET", FUNKSCHAU-Sonderheft Hobbycomputer 2.

Tabelle 2: Benutzte Interpreter-Routinen im ROM

B27F	Warmstart des Basic
B436	Suchen nach einer bestimmten Zeile, deren Nummer in14/15 gegeben ist
B76A	Y-Register wird zu C6/C7 addiert
B7DA	Umwandlung einer Dezimalzahl im Basic-Text in eine 16-bit-Zahl in 14/15
B900	Ausgabe einer CRLF-Folge
BB6B	Berechnung eines beliebigen numerischen Basic-Ausdruckes
BB6E	Überprüfung, ob Resultat numerisch
BB70	Überprüfung, ob Resultat ein String
BB7F	Berechnung eines beliebigen Ausdruckes
BCB9	Berechnung eines beliebigen, in Klammern stehenden Ausdrucks
BCBF	Test, ob C6/C7 auf eine Klammer ')' weist
BCC5	Test, ob C6/C7 auf ein Komma ',' weist
BCD0	Ausgabe von ?SN ERROR
BE6E	CY=1 wenn A ein Alpha-Zeichen (A...Z) enthält
BF87	Ausgabe von ?FC ERROR
C536	Umwandlung der Zahl imFloatingpoint-Akku in eine 16-bit-Zahl in 14/15 mit dem höherwertigen Byte in 15
C588	Zur Zahl im Floatingpoint-Akku (FAC) wird 0.5 addiert
C616	Normalisieren der Zahl in FAC
C767	FAC wird mit der Zahl multipliziert, auf die das Registerpaar A,Y weist. Die Zahl muß ebenfalls in Floatingpoint-Darstellung vorliegen
C821	FAC wird mit 10 multipliziert
C83D	FAC wird durch 10 dividiert
C8E1	Der Wert einer Variable, auf die das Registerpaar A,Y weist, wird nach FAC transferiert
C95A	Der Inhalt von FAC wird gerundet
CA0B	Ausführung der Funktion INT

Prüfsumme erkennt falsche Eingabe

Wenn man längere Programme aus Veröffentlichungen abtippt, so ergibt sich ganz automatisch eine gewisse Wahrscheinlichkeit, daß einige Bytes falsch übernommen wurden. Das Überprüfen kompletter Listings ist extrem zeitraubend und kostet viel Konzentration. Und auch dann ist es noch nicht auszuschließen, daß man einen Fehler ganz einfach übersieht.

Aus diesem Grunde führen wir bei längeren Maschinensprache-Programmen, soweit uns das technisch möglich ist, den Abdruck einer Prüfsumme ein. Diese Prüfsumme umfaßt vier Hex-Ziffern und entsteht durch Aufsummieren aller Bytes im Programm. Ein dabei entstehender Übertrag wird ignoriert.

Nach der Eingabe längerer Programme ist man also gut beraten, den Computer selbst erst einmal die Prüfsumme des jeweiligen Adressenbereiches errechnen zu lassen und mit dem im Heft abgedruckten Wert zu vergleichen.

Ein für die Mikrocomputer AIM-65 und PC-100 geeignetes 6502-Prüfsummen-Programm ist in *Bild 1* wiedergegeben. Nach dem Start mit Taste F1 kann man nach FROM= die Adresse des ersten und nach TO= diejenige Adresse eingeben, die dem letzten zu summierenden Byte folgt (d. h. Endadresse + 1). Will man z. B. die Summe des Bereiches 0200...02FF errechnen, so gibt man FROM=200 und TO=300 ein.

Bild 2 zeigt in der ersten Zeile die Prüfsumme des Programms in Bild 1 und in den darauffolgenden Zeilen die Prüfsummen unterschiedlicher Adressenbereiche der AIM-65/PC-100-Basic-Erweiterung von Michael Konz, die im gleichen Heft abgedruckt ist. Es ist empfehlenswert, erst einmal die Richtigkeit der Basic-Erweiterung auf diese Weise zu testen, bevor man sie an der Adresse 0220 startet, um zu vermeiden, daß sie wegen eines Eingabefehlers „Harakiri" läuft.

An dieser Stelle sei einigen Lesern gedankt, die die Idee zur Veröffentlichung von Prüfsummen bei Maschinenprogrammen an die Redaktion herantrugen.

Fe.

```
[K]*=0
/31
 0000 20 JSR E7A3
 0003 B0 BCS 0000
 0005 AD LDA A41C
 0008 85 STA F2
 000A AD LDA A41D
 000D 85 STA F3
 000F 20 JSR E7A7
 0012 B0 BCS 000F
 0014 A0 LDY #00
 0016 84 STY F0
 0018 84 STY F1
 001A 18 CLC
 001B B1 LDA (F2),Y
 001D 65 ADC F0
 001F 85 STA F0
 0021 90 BCC 0025
 0023 E6 INC F1
 0025 E6 INC F2
 0027 D0 BNE 002B
 0029 E6 INC F3
 002B A5 LDA F3
 002D CD CMP A41D
 0030 D0 BNE 001A
 0032 A5 LDA F2
 0034 CD CMP A41C
 0037 D0 BNE 001A
 0039 20 JSR E83E
 003C A5 LDA F1
 003E 20 JSR EA46
 0041 A5 LDA F0
 0043 4C JMP EA46

[M]=010C 4C 00 00 00
```

Bild 1. Programm zur Errechnung der Prüfsumme bestimmter Adressenbereiche für die Mikrocomputer AIM-65 und PC-100

```
[[]FROM=0TO=46 2C62
[[]FROM=220TO=300 71F1
[[]FROM=300TO=400 7C10
[[]FROM=400TO=500 7DAA
[[]FROM=500TO=600 86B2
[[]FROM=600TO=700 938C
[[]FROM=700TO=800 7630
[[]FROM=800TO=900 882C
[[]FROM=900TO=A2A 780F
[[]FROM=220TO=A2A FC54
```

Bild 2. Einige Prüfsummen-Beispiele: Oben für das Programm in Bild 1, darunter für die Basic-Erweiterung, die in diesem Heft für AIM-65 und PC-100 zu finden ist

Video-Treiber für den AIM-65

Der Anschluß eines Video-Interface mit serieller Schnittstelle (UART) an den TTY-Port der Mikrocomputer AIM-65 und PC-100 wurde bereits in FUNKSCHAU 1979, Heft 25, beschrieben. Aber: Das Drücken der DEL-Taste löschte das letzte Zeichen zwar auf dem LED-Display, nicht aber auf dem Bildschirm.

Die Applikationsschrift R6500N12 von Rockwell nennt eine Abhilfemöglichkeit: Wickelt man an den Anzeigezeiger A415 um die 19., so wird auch über die 20. Stelle hinaus noch das Delete-Zeichen der Tastatur geechot. Das hat zwar den Nachteil, daß das Display nach dem Erreichen der 20. Stelle nicht weiterrollt; wenn man mit dem Bildschirm arbeitet, spielt das aber kaum eine Rolle (*Bild*).

Befindet man sich in Basic, so ist darauf zu achten, daß man nicht über die 60. Spalte einer Zeile hinausgerät. Das könnte natürlich mit einem zusätzlichen Programmteil beseitigt werden, darauf wurde hier jedoch aus Gründen der Byte-Einsparung verzichtet. Leider funktioniert die Methode nur bei Basic, nicht aber mit dem Texteditor.

E. Brockt

```
[M]=0F90 00 00 00 48 8A 48 AE FF 0F E0 35 D0 03 CE FF 0F
[ ] 0FA0 AE 15 A4 E0 13 D0 06 CE 15 A4 EE FF 0F 68 AA 68
[ ] 0FB0 29 7F C9 0D F0 07 C9 20 F0 10 4C A8 EE 20 A8 EE
[ ] 0FC0 A9 00 8D FF 0F A9 0A 4C A8 EE AD 82 A4 C9 FD F0
[ ] 0FD0 05 A9 20 4C A8 EE AD FF 0F C9 00 F0 06 CE FF 0F
[ ] 0FE0 EE 15 A4 A9 08 4C A8 EE A9 93 8D 06 A4 A9 0F 8D
[ ] 0FF0 07 A4 A9 02 8D 17 A4 A9 FD 8D 18 A4 6C FC FF 00
[]]
[K]*=010C
/
 010C 4C JMP 0FE8
```

Hex-Dump des Video-Treibers. Die Bildschirm-Ausgabe läßt sich mit Taste F1 in Betrieb nehmen, wenn man bei 010C die Bytes 4C E8 0F einschreibt, und durch Druck auf Reset wieder ausschalten

µC-Nachrichten

In Hannover hat sich eine kleine SC/MP-II-Gruppe etabliert, die sich mit dem Prozessor INS 8086 beschäftigt. Interessenten können sich an Ing. Gerd Neumann, Pappelweg 20, 3000 Hannover 1, wenden.

F'

Wenn man an nichtflüchtige Speicher denkt, so meint man meist Magnetblasen oder EPROMs. Daß es auch anders geht, beweist ein CMOS-Speichermodul von Atron (Fritz-Meyer-Weg 55, 8 München 81). Es ist intern batteriegepuffert und läßt sich mit Mikrocomputern über eine parallele oder serielle Schnittstelle verbinden. Die Kapazität beträgt 4, 8 oder 16 KByte.

F'

Die Mainzer Firma Eltec, bekannt durch die Computer Eurocom-1 und -2 sowie durch recht preiswerte Video-Interface-Karten, bietet für den Eurocom-2 (CPU: 6809) nun Basic und Pascal als höhere Programmiersprachen an. Eltec war neben Motorola der erste Hersteller eines Computers mit dem Prozessor 6809, einer 8-bit-CPU mit interner 16-bit-Struktur.

Der Ideal-Computer

Eine Wunschliste an die Tischcomputer-Hersteller

Die heute erhältlichen Tischcomputer, Personal Computer, Heimcomputer oder wie man sie auch immer bezeichnen will, hätten vor zehn Jahren selbst optimistische Fachleute zum Staunen gebracht. Trotzdem: Hie und da bemerkt man noch eine gewisse Unausgereiftheit, die im folgenden aufs Korn genommen wird. Vielleicht denkt der eine oder andere Hersteller einmal über unseren Forderungs-Katalog nach; vielleicht orientiert sich auch der eine oder andere Kaufinteressent daran...

Bevor der Computer irgend etwas tut, muß man ihm über seine Tastatur erst einmal sagen, was er zu tun hat. Beginnen wir also mit diesem Teil des Tischcomputers.

In Deutschland besitzen Schreibmaschinen das Y gewöhnlich links unten, und außerdem findet man auf ihnen solche Zeichen wie Ä, Ö, Ü, ß. Es ist selbstverständlich, daß ein in Deutschland angebotener Computer die gleiche Tastatur wie so eine Schreibmaschine besitzen sollte.

Ebenso selbstverständlich ist es, daß die Tastatur eines Computers bei gedrückter Shift-Taste Groß-, bei nicht gedrückter Shift-Taste Kleinbuchstaben erzeugen sollte; heute ist das bei vielen Modellen noch genau umgekehrt. Um das Problem zu vermeiden, daß man dann bei der Eingabe von Programmbefehlen (die normalerweise mit Großbuchstaben geschrieben werden) dauernd Shift drücken muß, sollte eine Shift-Lock-Taste vorhanden sein, die auf Großschreibung „einrastet". Allerdings sollte „Shift Lock" nur für Buchstabentasten wirksam sein, nicht aber für die Ziffern- und Zeichentasten.

Recht praktisch ist auch eine Repeat-Taste, die dafür sorgt, daß eine gleichzeitig gedrückte Taste laufend in schneller Folge (ca. 10mal pro Sekunde) wiederholt wird. Eine Alternative ist eine automatische Repeat-Funktion, die wirksam wird, wenn man eine Taste länger als etwa eine Sekunde gedrückt hält. (Es gibt Computer, die zwar eine mit „Repeat" beschriftete Taste besitzen, die aber ohne erst zusätzlich zu ladende Software die von ihr zu erwartende Funktion nicht ausführt!)

Daß die Tastatur nicht prellen und überhaupt die mechanische Qualität einer guten Schreibmaschinen-Tastatur haben sollte, ist leider ebenfalls noch nicht für alle Hersteller selbstverständlich.

Die Anzeige

Während heute noch ein Bildschirm als Standard-Anzeige dient, kann sich das schon in ein paar Jahren ändern: Der flache Bildschirm, sprich das Festkörper- oder LCD-Matrix-Display, ist schon heute für die Darstellung von Schriftzeichen realisierbar, wenn auch noch ziemlich teuer.

Unabhängig von der Anzeigen-Technologie ist es wünschenswert, wenigstens 16 Zeilen mit je 64 Zeichen darzustellen. Was wir von der Tastatur fordern, gilt auch für die Anzeige: Deutsche Sonderzeichen wie ä, ö, ü, ß sollten Standard sein. Um eine gute Lesbarkeit zu gewährleisten, empfiehlt sich eine Zeichendarstellung als 7x9-Punktmatrix (heute ist eine 5x7-Matrix noch Standard) mit „Unterlängen" bei den Buchstaben g, j, p, q und y.

Legt man den heutigen Standard beim „Bildschirmtext"-System zugrunde und ermöglicht das Aufteilen eines Zeichenquadrates in 2x3 Grafikquadrate, so ergibt sich eine Auflösung bei grafischen Darstellungen (Kurven u. a.) von wenigstens 48 Punkten vertikal und 128 Punkten horizontal. Es wäre sehr sinnvoll, wenn sich die Hersteller auf diese (oder zumindest auf eine einheitliche andere) Grafik-Norm einigen könnten, um einen Programmaustausch zwischen unterschiedlichen Computertypen zu ermöglichen.

Eine sehr praktische Sache ist das „Screen Editing". Darunter versteht man, daß man mit dem Cursor ein auf dem Bildschirm stehendes Programm unmittelbar redigieren kann; der Bildschirminhalt wird also direkt in den Speicher übernommen (z. B. bei PET-2001, CBM-3001). Diese Eigenschaft will man schon bald nicht mehr missen.

„Screen Editing" setzt im Computer hardwaremäßig voraus, daß das Video-RAM vom Mikroprozessor aus wie ein normaler RAM-Bereich adressiert werden kann. Selbstverständlich kann auch der Benutzer dann z. B. mit PEEK und POKE direkt auf jede Stelle des Bildschirms zugreifen und Zeichen an beliebigen Positionen schreiben. Dabei ist es sehr störend, wenn die verwendete interne Zeichencodierung nicht mit der ISO-7-bit-Norm bzw. ASCII übereinstimmt.

Wünschenswert ist also, daß die intern verwendeten Zeichencodes normgerecht sind, z. B. dezimal 65 für ein großes A und 97 für ein kleines a.

Das Betriebssystem

Unter Betriebssystem verstehen wir jene intern im Computer vorhandenen Programme, die dem Benutzer die Kommunikation mit dem Rechner erlauben. Dazu gehört auch der Basic-Interpreter. Hier gibt es zwei Möglichkeiten. Entweder ist das Betriebssystem fest in ROMs gespeichert und steht daher sofort beim Einschalten zur Verfügung, oder es muß beim Einschalten erst geladen werden, was dann möglichst einfach und schnell gehen sollte. Die zweite Methode hat den manchmal bedeutsamen Vorteil, daß man den Computer leichter auf eine andere Programmiersprache (z. B. Pascal, Fortran usw.) umrüsten kann. Wenn man nur in Basic arbeiten möchte, wird man dagegen ein in ROMs gespeichertes Betriebssystem vorziehen. Auch in Zukunft wird es also berechtigt sein, wenn es Computer mit beiden Versionen gibt.

Keine Diskussion sollte es dagegen bei der Dokumentation dieses Betriebssystems geben. Der Anwender sollte zumindest auf Anforderung ein vollständiges, kommentiertes Assembler-Listing bzw. eine Speicherbelegungs-Liste mit Nennung aller verwendbaren ROM-Unterprogramme erhalten können. Das bedeutet, daß der Computerhersteller mit dem meist nicht im eigenen Haus befindlichen Entwickler des Basic-Interpreters vereinbaren muß, daß die Herausgabe eines solchen Listings urheberrechtlich zulässig ist.

Die Hersteller sollten sich endlich auf eine einheitliche Befehls-Syntax einigen. Es ist für einen Anwender unverständlich, warum einzelne Hersteller für Befehle, deren Wirkung völlig identisch ist, unterschiedliche Worte verwenden müssen (z. B. A$=INKEY$ bei Tandy und GET A$ bei den meisten anderen Computern).

Ab und zu hängen sich Computer auf. Das bedeutet, daß sie wegen eventueller Programmierfehler in Maschinensprache, aber auch wegen Störimpulsen aus dem Netz jede weitere Befehlsannahme verweigern und sich nur noch durch Aus- und Einschalten aus dem „Weltraum" zurückholen lassen – natürlich sind alle gespeicherten Daten und das Programm damit gelöscht (z. B. PET, CBM).

Nicht viel besser ist es, wenn der Computer zwar über Reset-Taste verfügt, deren Betätigung aber wiederum zur Löschung des gespeicherten Basic-Programms sowie aller Variablenwerte führt (z. B. TRS-80, DAI).

Eine schon brauchbarere Lösung enthalten z. B. AIM-65, Superboard und PC-100. Nach dem Drücken der Reset-Taste kann man über eine „Warmstart"-Taste in den Basic-Interpreter zurück, ohne das Programm dabei zu löschen. Solange man das Basic-Programm dann nicht mit RUN wieder startet, kann man alle Variablenwerte zumindest noch von Hand abrufen.

Eigentlich selbstverständlich sollte die Möglichkeit sein, mit PEEK, POKE, USR und SYS eine Verbindung zwischen Basic-Interpreter und Maschinenebene herzustellen, um zeitkritische Aufgaben in Assemblersprache programmieren und von Basic-Programmen her aufrufen zu können.

Und noch etwas: Beim Auflisten von Programmen (LIST in Basic) sollte man z. B. durch Niederhalten einer Taste die Möglichkeit haben, die Ausgabe auf dem Bildschirm bzw. auf der Anzeige zu verlangsamen. Paradoxerweise ist diese Möglichkeit bei den Computern, die das wegen ihres einzeiligen Displays am nötigsten hätten (AIM-65 und PC-100), nur mit zusätzlichen Maschinenprogrammen möglich.

Schnittstellen

Ganz selbstverständlich sollte der Computer seinen gesamten Systembus (Adressen-, Daten-, Steuerleitungen) an eine Steckleiste herausgeführt haben, um dem Anwender spätere Systemerweiterungen zu gestatten. Diese Forderung wird schon heute meist erfüllt.

Etwas spärlicher sieht es dagegen oft mit anderen Möglichkeiten aus. Externes auszuschließen. Wünschenswert wäre wenigstens ein 8-bit-Port, acht bidirektionale Leitungen also, an die man Schalter, LEDs, Relais und anderes zumindest über einfache Treibertransistoren anschließen kann. Diesen Port kann der Benutzer dann selbst so programmieren, wie es sein Anwendungsfall erfordert – er kann daraus sogar eine serielle Schnittstelle machen.

Nützlich ist natürlich, wenn eine solche serielle Schnittstelle (V-24, RS-232, 20-mA-Stromschleife) schon eingebaut ist. Zu fordern ist allerdings, daß z. B. über ein programmierbares „Flag" alle INPUT-, PRINT-, LOAD- und SAVE-Operationen auf diese Schnittstelle umgeleitet und alle üblichen Baudraten (110, 150, 300, 600, 1200 usw.) programmiert werden können. Auch hier können AIM-65 und PC-100 als Musterbeispiel dienen.

Der IEC-Bus hat zwar bei Meßaufgaben durchaus seine Berechtigung, ist aber für die Verbindung von Computer mit Drucker und Floppy vor allem wegen seiner begrenzten Transfergeschwindigkeit nicht unbedingt optimal. Hier kann es günstiger sein, die Floppy-Disk direkt über den Adressen- und Datenbus zugreifen zu lassen.

Externe Speichermedien

Apropos Floppy-Disk: Lobenswerterweise passen sich manche Hersteller bereits dem IBM-Aufzeichnungsstandard an, um die Austauschbarkeit von Software zu gewährleisten. Dazu ist aber auch erforderlich, das Datenformat selbst (d. h. z. B. Anfangs- und Endkennzeichnung, Codierung der Zeichen usw.) zu vereinheitlichen.

Viele Computer zeichnen Basic-Programme nicht Buchstaben für Buchstaben, Zeichen für Zeichen auf, sondern fassen Basic-Befehlsworte zu einem einzigen Byte zusammen. Das führt wiederum zur Inkompatibilität von Disketten (und auch Kassetten) mit Basic-Programmen bei den unterschiedlichen Computertypen. Die einfachste Lösung dieses Problems ist es, diese Kurzkennzeichen für Basic-Befehle zwar intern im Arbeitsspeicher, nicht aber für das Abspeichern auf Floppy oder Diskette zu verwenden (wie bei AIM-65, PC-100, Superboard). Das führt natürlich zu etwas längeren Zeiten für Abspeichern und Laden.

Wenn ein Hersteller sich deshalb nicht von der „komprimierten" Aufzeichnung trennen kann, so sollte er wenigstens dem Anwender ein zweites, mit anderen Typen kompatibles Aufzeichnungsformat zusätzlich zur Verfügung stellen. Der ROM-Speicherbedarf dafür ist vergleichweise gering und wird mit Sicherheit vom Käufer honoriert. Bleibt zu hoffen, daß die Hersteller das einsehen und vom kleinkarierten „Ich-koche-meine-eigene-Suppe"-Denken wegkommen!

Kurz gesagt:

- Deutsche Tastatur mit ä, ö, ü, ß
- „Richtige" Groß- und Kleinschreibung
- Dokumentation des Betriebssystems
- Einheitliche Syntax für gleichwertige Befehle
- Genormte Schnittstellen plus Parallel-Port
- Mit Fremdfabrikaten kompatibles Datenformat für Kassetten und Floppy

Was noch bleibt

Was noch zu sagen bleibt, sind Kleinigkeiten, und eigentlich ist es beschämend, daß man so etwas überhaupt erwähnen muß, aber manch heute existierender Tischcomputer macht das unumgänglich. Dazu gehört es zum Beispiel, daß es dem Gerät nichts ausmachen darf, wenn es bei Umgebungstemperaturen um 30 °C im Dauerbetrieb läuft; daß man Programme auf Kassette mit vernünftigen Namen abspeichern und wieder laden kann; daß Fehler beim Laden von Programmen zuverlässig erkannt und gemeldet werden und daß der Computer keine unzumutbare Störstrahlung produziert, was man leicht mit einem danebengestellten Kofferradio überprüfen kann.

Wenn sich auch die meisten Hersteller im fernen Amerika nicht um die vorstehenden Zeilen scheren werden, so finden doch zumindest Sie, verehrte Leser, darin vielleicht die eine oder andere Anregung, wonach Sie beim Kauf eines Tischcomputers fragen können.

Herwig Feichtinger

Funkfern-schreiben mit AIM-65 und PC-100

Das im folgenden Beitrag beschriebene 6502-Maschinenprogramm eignet sich gleichermaßen für die Mikrocomputer AIM-65 (4 KByte) von Rockwell und PC-100 von Siemens. Es erlaubt das Empfangen und Senden von Funkfernschreib-Signalen im Baudot-Code und besorgt auch die postalisch vorgeschriebene Ausstrahlung des Stations-Rufzeichens per Sprachsynthese. Damit ist es käuflichen Fernschreibcomputern weit überlegen.

RTTY-Programme für die CPU 6502 wurden schon mehrmals veröffentlicht [1, 2, 3, 4, 5, 7]. Die hier vorgestellte Version hebt sich aber durch einige deutliche Verbesserungen von ihren Vorläufern ab:

1. Bei Senden und Empfang ist es möglich, den Thermodrucker mitlaufen zu lassen. Ein Pufferspeicher sorgt dafür, daß dabei keine Zeichen verlorengehen [7].
2. Sender-Tonfrequenzen (Mark und Space), Baudraten und andere Betriebsparameter lassen sich nun durch einfachen Tastendruck während Senden und Empfangen ändern.
3. Es ist möglich, praktisch beliebig viele Standard-Texte mit dem AIM-Texteditor zu speichern und auf Tastendruck zu senden, z. B. CQ-Ruf, Stationsvorstellung, Test-Zeichenfolge usw.
4. Während des Empfanges ist es möglich, einen Text mit dem Editor zum Senden vorzubereiten. Währenddessen werden bis zu 255 empfangene Zeichen zwischengespeichert und ausgedruckt, sobald man vom Editor ins Monitorprogramm springt (Escape) und zum Empfangsprogramm zurückgeht (F2).

All dies wurde möglich, indem ausgiebige Verwendung von den Interrupt-Möglichkeiten des VIA-Bausteines 6522 gemacht wurde. Das Empfangen und Senden von Zeichen erfolgt völlig unabhängig vom parallel laufenden Hauptprogramm über den Pufferspeicher (FIFO). Deshalb ist es sogar möglich, das Hauptprogramm zu verlassen, im Monitorprogramm „herumzufuhrwerken" und beim Zurückgehen zum Warmstart des Empfangsprogrammes die in der Zwischenzeit eingegangenen Zeichen auszudrucken.

Die im Programm implementierte Sprachsynthese zur Rufzeichenausstrahlung ähnelt derjenigen, die bereits im FUNKSCHAU-Sonderheft „Mikrocomputer-Anwendungen" veröffentlicht wurde [6], verzichtet aber auf die Redundanzverminderung in den Sprachpausen, da solche während des gesprochenen Rufzeichens ohnehin kaum auftreten. Deshalb kann zur Aufnahme auch ein vereinfachtes Digitalisierungs-Programm verwendet werden (*Bild 1*).

Inbetriebnahme

Das vollständige RTTY-Programm ist in *Bild 2* als Hex-Dump wiedergegeben. Leider war aus Platzgründen der Abdruck eines kommentierten Assembler-Listings nicht möglich, da dies allein schon rund fünf Seiten gekostet hätte. Alternativ nennt deshalb *Tabelle 1* die wichtigsten Adressen und Programmteile, die es zusammen mit einem Disassembler-Ausdruck leicht machen, die RTTY-Software zu analysieren.

Bei der Inbetriebnahme empfiehlt es sich, beim Eingeben der Programmteile folgendermaßen vorzugehen:

1. Das Spracheingabe-Programm aus Bild 1 eingeben und zunächst auf Kassette abspeichern (Adressenbereiche: 010C...0111 und 0F00 bis 0F60).
2. Das RTTY-Programm aus Bild 2 eingeben und ebenfalls sicherheitshalber zunächst ebenfalls auf Kassette abspeichern (Adressenbereiche 010C...0114 und 0200...053E).
3. Drückt man nun F1 (vom Monitorprogramm aus), so muß bereits der Empfang möglich sein, wobei am Port PA 0 ein RTTY-Konverter angeschlossen sein muß; die Polarität sollte dabei so liegen, daß ein empfangenes Mark-Signal Low-Pegel an PA 0 entspricht. Nach Druck auf Y muß auch das Senden von eingetippten Texten möglich sein.

Alles, was bis hierher noch nicht funktioniert, ist erstens die Rufzeichen-Sprachsynthese, statt derer nur ein undefinierbares Geräusch zu hören ist, und zweitens das Abrufen von Standardtexten. Beim Eintippen von Texten kann auf „Return" verzichtet werden: Das Programm sorgt am Ende einer Zeile von selbst für das Einfügen einer

```
[M]=0F00 A9 37 8D 02 A8 A9 EE 8D 0C A8 20 4D 0F A2 08 48
[ ] 0F10 20 56 0F AD 00 A8 0A 68 6A CA D0 F3 81 FE 20 42
[ ] 0F20 0F D0 EA 60 A9 80 8D 02 A0 20 4D 0F A2 00 A1 FE
[ ] 0F30 A2 08 20 56 0F 6A 6E 00 A0 CA D0 F6 20 42 0F D0
[ ] 0F40 ED 60 E6 FE D0 02 E6 FF A5 FF C9 0D 60 A9 3F 85
[ ] 0F50 FE A9 05 85 FF 60 2C 85 A4 10 FB A0 60 8C 94 A4
[ ] 0F60 60 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
[*]=010C

[/] 010C 4C 00 0F 4C
[/] 0110 24 0F
```

Bild 1. Dieses Programm wird nur vorübergehend benötigt. Es dient zur Digitalisierung des Sprach-Rufzeichens, um es vom Band in den Speicher zu übernehmen. Vorsicht: Dieser Adressenbereich wird bei laufendem RTTY-Programm vom Pufferspeicher überschrieben!

Wagenrücklauf/Zeilenvorschub-Folge, sobald ein Leerraum oder Bindestrich erkannt wird, so daß keine Worte „zerrissen" werden. (Das AIM-Monitorprogramm begrenzt die Zeilenbreite bei Senden und Empfang auf max. 60 Zeichen.)

Eingabe von Standardtexten

Hat man das RTTY-Programm bis hierher zum Laufen gebracht, so kann man den AIM-Texteditor vom Monitorprogramm aus für den Adressenbereich FROM=0D00; TO=0F00 initialisieren (4-KByte-AIM). Hat man eine Speichererweiterung angeschlossen, die im Adressenbereich über 1000 liegt, ist eine Initialisierung auch dafür möglich.

Nun kann man Zeile für Zeile die Standardtexte eingeben. Jeder beginnt mit dem Doppelkreuz-Zeichen # und endet auch entweder mit diesem oder dem Ende des Texteditors, das intern als Null-Byte geführt wird. Will man z.B. als Standardtext „R" die Testbitfolge RYRYRY... und als Standardtext „T" die Folge „THE QUCK BROWN FOX..." eingeben, so hat das wie folgt zu geschehen:

#RRYRYRYRY...
#TTHE QUICK BROWN...

Dabei wird das erste dem Doppelkreuz folgende Zeichen jeweils zur Identifizierung eines Standardtextes verwendet und ebenso wie das Doppelkreuz selbst nicht mit ausgedruckt.

Ein Standardtext kann auch ohne weiteres aus mehreren Textzeilen bestehen, da zur Trennung der Texte voneinander nicht das Return-Zeichen (hex 0D) dient, sondern ausschließlich das Doppelkreuz. Hat man die gewünschten Texte eingegeben – sie dürfen beim 4-K-AIM maximal 512 Zeichen umfassen – so kann man sie getrennt oder gemeinsam mit dem RTTY-Programm auf Kassette abspeichern; die dabei zu verwendenden Adressenbereiche sind 00DF...00E6 und 0D00...0EFF.

```
[M]=0200 A2 FF 9A 20 7A 03 20 56 04 A9 C0 8D 0B A0 A2 14
[ ] 0210 8E 05 A0 86 00 A9 1F 20 7B 02 CA D0 F8 20 43 02
[ ] 0220 20 83 FE 20 3B 04 90 F8 20 2E 02 4C 20 02 24 11
[ ] 0230 10 0D C9 20 F0 04 C9 2D D0 05 20 49 02 D0 04 C9
[ ] 0240 0D D0 06 A9 0D A4 07 84 11 20 BC E9 C6 11 A0 00
[ ] 0250 D9 80 04 F0 05 C8 C0 3F D0 F6 98 48 29 20 C5 12
[ ] 0260 F0 0F 85 12 C9 00 F0 04 A9 1B D0 02 A9 1F 20 7B
[ ] 0270 02 68 C9 08 D0 05 20 7B 02 A9 02 A4 0C 99 00 0F
[ ] 0280 C8 84 0C C8 C4 0D F0 FC 60 24 00 10 4A 48 98 48
[ ] 0290 A5 0E F0 1A C6 0E F0 2A 18 A9 01 2D 01 A0 45 01
[ ] 02A0 D0 01 38 66 0F A5 06 8D 09 A0 68 A8 68 40 A9 01
[ ] 02B0 2D 01 A0 45 01 F0 1C A9 06 85 0E A5 06 4A 65 06
[ ] 02C0 D0 E5 A5 0F 4A 4A 4A F0 DC E6 0C A4 0C 99 00 0F
[ ] 02D0 4C A5 02 A9 02 D0 D0 48 A5 06 8D 09 A0 A5 0E F0
[ ] 02E0 1D 38 66 0F 90 0A A5 02 8D 04 A0 A5 03 4C F7 02
[ ] 02F0 A5 04 8D 04 A0 A5 05 8D 07 A0 C6 0E 10 25 A9 08
[ ] 0300 85 0E 86 10 A6 0D E4 0C D0 0A 24 08 10 13 A9 FE
[ ] 0310 85 12 D0 0B BD 00 0F F0 17 0A 09 C0 E8 86 0D 85
[ ] 0320 0F A6 10 68 40 A2 07 BD 37 05 95 01 CA 10 F8 D8
[ ] 0330 A2 FF 86 00 9A E8 8E 0B A0 20 7A 03 20 56 04 A9
[ ] 0340 00 85 12 A4 0D C4 0C F0 20 E6 0D B9 00 0F F0 EF
[ ] 0350 C9 1F F0 EB C9 1B D0 04 A9 20 D0 E5 05 12 AA BD
[ ] 0360 80 04 C9 40 F0 D9 20 BC E9 20 EF EC 98 F0 D4 20
[ ] 0370 43 EC 29 1F 20 3B 04 4C 43 03 A9 89 8D 04 A4 A9
[ ] 0380 02 8D 05 A4 A9 80 8D 02 A0 A9 A0 8D 0E A0 8D 09
[ ] 0390 A0 A2 00 86 0C 86 0D 58 60 A4 E3 84 09 A4 E4 84
[ ] 03A0 0A 4C 43 02 E6 09 D0 02 E6 0A A0 00 60 A2 3C 20
[ ] 03B0 DA 03 85 0B 20 99 03 A0 00 B1 09 F0 38 20 A4 03
[ ] 03C0 C9 23 D0 F3 B1 09 C5 0B D0 ED 20 A4 03 B1 09 F0
[ ] 03D0 24 C9 23 F0 20 20 2E 02 D0 F0 20 F0 E9 BD E2 04
[ ] 03E0 F0 06 20 BC E9 E8 D0 F5 20 83 FE 20 BC E9 48 29
[ ] 03F0 03 AA 0A A8 68 60 A2 13 20 DA 03 B9 DA 04 85 02

[M]=0400 B9 DB 04 85 03 A2 1F 20 DA 03 B9 DA 04 85 04 B9
[ ] 0410 DB 04 85 05 60 A2 2C 20 DA 03 86 01 60 18 90 01
[ ] 0420 38 66 08 60 A2 00 20 DA 03 BD D6 04 85 06 60 A2
[ ] 0430 43 20 DA 03 BD D4 04 85 07 38 60 C9 1B B0 FA C9
[ ] 0440 10 90 F6 E9 10 0A AA BD BF 04 85 09 BD C0 04 85
[ ] 0450 0A 20 7D 04 18 60 A9 3F 85 09 A9 05 85 0A A1 09
[ ] 0460 A2 08 2C 85 A4 10 FB A0 60 8C 94 A4 6A 6E 00 A0
[ ] 0470 CA D0 EF 20 A4 03 A5 0A C9 0D D0 E2 60 6C 09 00
[ ] 0480 5D 45 40 41 20 53 49 55 0D 44 52 4A 4E 46 43 4B
[ ] 0490 54 5A 4C 57 48 59 50 51 4F 42 47 40 4D 58 56 40
[ ] 04A0 40 33 40 2D 20 27 38 37 0D 40 34 3B 2C 5B 3A 28
[ ] 04B0 35 2B 29 32 21 36 30 31 39 3F 40 40 2E 2F 3D E1
[ ] 04C0 E6 F6 03 15 04 AD 03 CF F6 20 04 24 04 2F 04 30
[ ] 04D0 03 00 02 1D 04 21 32 54 4D 32 E9 00 86 01 28 01
[ ] 04E0 FD 00 31 3D 34 35 20 32 3D 35 30 20 33 3D 37 35
[ ] 04F0 20 42 44 20 00 4D 41 52 4B 20 31 2D 34 28 34 29
[ ] 0500 00 53 50 41 43 45 20 31 2D 34 28 31 29 00 52 58
[ ] 0510 2D 4D 41 52 4B 20 30 2F 31 3F 28 30 29 00 54 45
[ ] 0520 58 54 23 3F 00 57 49 44 54 48 3A 20 31 3D 34 30
[ ] 0530 20 32 3D 36 30 20 00 00 E9 00 86 01 54 32 80 AA

[/] 010C 4C 25 03 4C
[/] 0110 43 03 4C 20
[/] 0114 02

FROM=0200 TO=0300  5FDF
FROM=0300 TO=0400  6EF6
FROM=0400 TO=0500  52A4
FROM=0500 TO=053F  0DF0
FROM=0200 TO=053F  2F69
```

Bild 2. Das komplette RTTY-Programm umfaßt weniger als 1 KByte. Um Fehler beim Abtippen des Hex-Dump erkennen zu können, ist für unterschiedliche Adressenbereiche noch eine Prüfsumme der Datenbytes angegeben, wie dies in Heft 2/1981 auf Seite 84 erläutert ist

Sprachsynthese-Rufzeichen

Jetzt können wir uns endlich der Eingabe des eigenen Rufzeichens als digitalisierte Sprache zuwenden. Dazu laden wir das Programm aus Bild 1 und sprechen unser Rufzeichen mit einem geeigneten Mikrofon in möglichst geräuscharmer Umgebung sowie möglichst prägnant und verständlich auf Kassette.

Für das Einlesen in den Speicher sollte dem Widerstand R18 auf der AIM-Platine von 5,1 MΩ ein solcher von 390 kΩ parallelgeschaltet werden. Durch Druck auf F1 wird nun rund eine Sekunde lang die aufgenommene Sprache von der Kassette in den Speicher des AIM/PC-100 übernommen – man muß es u.U. ein paarmal versuchen, bis man das Rufzeichen komplett sauber im Speicher hat und es von 053F...0CFF auf Kassette abspeichern kann. Ein Druck auf F2 sorgt dafür, daß man sich das Gespeicherte anhören kann, indem man einen Verstärker oder hochohmigen Kopfhörer an das Interface in *Bild 3* anschließt, das sowohl einen einfachen Empfangskonverter für UKW als auch die Anpassung an den Sendereingang enthält.

Das Hardware-Interface

Da die Töne zur Modulation des Senders direkt vom VIA-Baustein im freilaufenden Timer-Modus erzeugt werden, hält sich der nötige Hardware-

Aufwand zum Anschluß des Mikrocomputer an einen UKW- oder KW-Transceiver in Grenzen.

Die Schaltung in Bild 3 enthält daher lediglich einen PLL-Demodulator, der hier auf 2125 Hz (Mark in der Normallage) abgestimmt wird und dem zum Ausgleich von Nf-Pegelschwankungen ein einfacher Begrenzerverstärker vorangestellt ist, sowie (sendeseitig) einen Tiefpaß, der sowohl die erzeugte Sprache als auch das zunächst rechteckförmige Nf-Ausgangssignal von unerwünschten Oberwellen säubert.

Sein Ausgangssignal kann direkt dem Mikrofoneingang des Transceivers zugeleitet werden. Dabei sollte man den Pegel so einstellen, daß der Sender nicht voll ausgefahren wird, um die erneute Bildung von Nf-Harmonischen innerhalb des Modulators zu vermeiden. Ferner sind manche SSB-Sender thermisch nicht für den RTTY-Dauerstrichbetrieb bei maximaler Ausgangsleistung geeignet.

Die Steuerung der Sende/Empfangsumschaltung kann zwar von Hand erfolgen; noch einfacher und bequemer ist es aber, dies dem Transceiver selbst durch Einschalten der heute meist vorhanden „VOX" zu überlassen.

Jetzt ist alles komplett

Hat man alle Programmteile einzeln auf Kassette gebracht, so lädt man sie nun nacheinander in den Speicher (außer dem Spracheingabe-Programm in Bild 1, das vorerst nicht mehr benötigt wird) und fertigt auf Kassette eine Komplettversion an, die aus den Adressenbereichen 010C...0114, 00DF...00E6 und 0200...0EFF besteht.

Tabelle 2 zeigt nun, wozu das RTTY-Programm fähig ist. Vom Monitorprogramm aus wird es mit der Taste F1 gestartet und ist dann – nachdem noch das Rufzeichen „gesagt" wird – im Empfangsmodus (45 Bd). Durch kurzen Druck auf P kann man den Thermodrucker ein- und ausschalten; dabei wird im Display ON oder OFF eingeblendet.

Drückt man V, so erscheint im Display „1=45, 2=50, 3=75 BD", und durch anschließenden Druck auf 1, 2 oder 3 kann man die Baudrate ändern. Ähnlich funktioniert das bei den meisten anderen in Tabelle 2 genannten Tastenbefehlen.

Ein Druck auf Y schaltet in den Sendemodus; das Programm strahlt dann zunächst das Sprach-Rufzeichen aus; anschließend folgen 20 BU-Umschaltzeichen und dann der eingetippte Text, wobei man bis zu 255 Baudot-Zeichen vorausschreiben kann.

Die in Tabelle 2 genannten Tastenbefehle sind während des Sendens nur durch gleichzeitiges Drücken von CTRL zugänglich, z.B. ruft man einen Standardtext mit CTRL S ab. Dann erscheint im Display „TEXT#?", wonach man den jeweiligen Text-Kennbuchstaben drücken muß (im obigen Beispiel ein R zum Ausstrahlen einer RYRYRY-Folge). Der Drucker läßt sich sowohl mit CTRL PRINT als auch mit CTRL P ein- und ausschalten. Als letztes Zeichen des zu sendenden Textes drückt man F2. Das resultierende Zeichen (eckige Klammer) wird nicht mehr mit ausgestrahlt, sondern sorgt dafür, daß das Programm automatisch in den Empfangsmodus zurückspringt, sobald der Text komplett gesendet ist – natürlich wird vorher noch das Rufzeichen „gesagt".

Im Empfangsmodus kann man den Ausdruck empfangener Zeichen durch Drücken der Space-Taste anhalten; läßt man sie wieder los, so werden die in der Zwischenzeit eingegangenen Zeichen „nachgeholt", indem sie aus dem Pufferspeicher geholt werden. Das Antippen der CTRL- oder Shift-Tasten hat eine ähnliche Wirkung, jedoch wird beim Loslassen der Ausdruck noch

Tabelle 1:
Wichtige Adressen und Programmteile

Adresse	Bedeutung
1. Zero-Page-Adressen	
0000	Sende/Empfangs-Flag
0001	Flag für RX-Konverter-Polarität
0002,0003	Mark-Sendefrequenz
0004,0005	Space-Sendefrequenz
0006	Baudrate
0007	Sender-Zeilenbreite
0008	Flag für BU-Füllzeichen
0009,000A	Pointer
000B	Standardtext-Code
000C	FIFO-Pointer 1
000D	FIFO-Pointer 2
000E	Bitzähler
000F	Zeichenpuffer
0010	X-Zwischenspeicher
0011	Zeichenzähler für Zeilenbreite
0012	BU/ZI-Flag (00/20)
2. Systemadressen	
A000-A00F	VIA 6522 (Ports und Timer)
A494	6532-Timer (DIV 64)
A485	6532-Timer-Flag (Bit 7)
FE83	Einlesen eines Zeichens von der Tastatur m. Display
ECEF	Test, ob eine Taste gedrückt ist
EC43	Tastatur-Zeichen holen, ohne Display
E9BC	Zeichen auf Display/Drucker ausgeben
F6CF	Warmstartadresse des Texteditors
E9F0	CR/LF an Display/Drucker ausgeben
E6E1	Drucker ein/ausschalten
A404,A405	IRQ-Vektor (im RAM)
00E3,00E4	Zeiger zum Beginn des Editor-Textes
3. Adressen im RTTY-Programm	
0200	TX-Kaltstart
0215	20 BU-Zeichen senden
0220-022D	TX-Hauptschleife
022E	Zeichen senden, ggf. CRLF einfügen
0250	Umwandlung ASCII in Baudot
0284	Warten, falls FIFO voll
0289	IRQ-Routine
028D	RX-Interrupt-Routine
02D7	TX-Interrupt-Routine
0325	Kaltstart (Taste F1), Betriebsparameter initialisieren
0330	RX-Kaltstart
0343-0379	RX-Hauptschleife
037A	VIA, Vektoren und Pointer initialisieren
0399	Pointer auf Beginn des Editortextes setzen
03A4	Pointer um 1 erhöhen
03AD	CTRL S ausführen
03DA	Prompting-Text ausgeben
03F6	CTRL Q ausführen
0415	CTRL R ausführen
041D	CTRL Z ausführen
0420,0424	CTRL U, CTRL V ausführen
042F	CTRL W ausführen
043B	Akkuinhalt auf CTRL-Zeichen überprüfen und ggf. Steuerbefehl ausführen
0456	Sprach-Rufzeichen ausgeben
047D	Indirekter Sprung via Pointer für Steuerbefehle
0480-04BE	Baudot-ASCII-Umwandlungstabelle
04BF-04D4	Sprungadressen für CTRL P bis CTRL Z
04D5-04D6	Tabelle für Zeilenbreite
04D7-04D9	Baudraten-Tabelle
04DA-04E1	Tabelle mit Tonfrequenzen
04E2-0536	ASCII-Tabelle mit Prompting-Texten
0537-053E	Initialisierungswerte für 0001...0008
053F-0CFF	Digitalisiertes Sprach-Rufzeichen
0D00-0EFF	Speicher für Editor-Texte
0F00-0FFF	FIFO (Pufferspeicher für RX und TX)

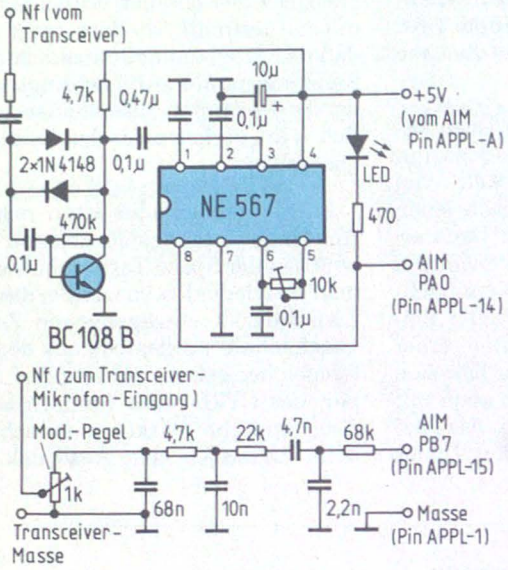


Bild 3. Wenn nicht ohnehin schon ein Empfangskonverter mit TTL-Ausgang vorhanden ist, kann diese Schaltung verwendet werden; sie arbeitet mit einem PLL-Decoder. Sendeseitig ist nur ein Tiefpaß vorhanden, um Oberwellen der Modulationsfrequenz zu unterdrücken

nicht fortgesetzt, sondern erst dann, wenn man wiederum kurz auf die Space-Taste tippt.

Editieren von Texten

Sowohl aus dem Sendeprogramm als auch – und das ist der Clou – aus dem Empfangsprogramm kann man mit CTRL T bzw. T zum Texteditor gehen; Sendung bzw. Empfang von Zeichen werden dabei nicht unterbrochen! Also ist es möglich, während des Empfanges im Texteditor einen Text vorzubereiten, mit Escape zum Monitor und mit F2 in den Empfangsmodus zurückgehen. Ist der Drucker dabei eingeschaltet, so holt er automatisch alle Zeichen nach (max. 255), die in der Zwischenzeit empfangen wurden. Geht man dann mit Y in den Sendemodus, so kann man mit CTRL S den vorbereiteten Text wie einen Standardtext aussenden – ein Komfort, den man sonst nur bei relativ teuren Funkfernschreib-Terminals findet, die aber wiederum nicht über die komfortablen Editiermöglichkeiten des AIM-Texteditors wie „Change" und „Find" verfügen. Man darf also, während Zeichen empfangen oder gesendet werden, ziemlich alles mögliche tun – nur eines nicht: mit dem Kassettenrecorder arbeiten. Die Kassettenroutinen schalten nämlich per SEI-Befehl den Interrupt ab, so daß Sendung bzw. Empfang unterbrochen würde.

Wie man Texte in den Editor eingibt oder sie verändert, ist so ausführlich in den Systemhandbüchern von AIM-65 und PC-100 beschrieben, daß hier darauf nicht näher eingegangen werden muß. Die Tatsache, daß der AIM ein solches Editor-Programm bereits im ROM enthält, hat das RTTY-Programm natürlich erheblich vereinfacht, führt aber leider auch dazu, daß der Editor-Komfort sich nicht ohne weiteres übernehmen läßt, wenn man versucht, das RTTY-Programm auf andere 6502-Computer umzuschreiben.

Herwig Feichtinger (DC1YB)

Tabelle 2: Tastatur-Steuerbefehle

Taste (TX: CTRL)	Wirkung
P	Drucker ein/ausschalten
Q	Mark/Space-Frequenzen eingeben
R	RX-Konverter-Polarität ändern
S	Standardtext aus dem Editor senden
T	Zum Texteditor springen
U	BU-Füllzeichen einschalten
V	Baudrate eingeben
W	Zeilenformatierung ändern (f. autom. Zeilenvorschub bei Sendung)
X	Sprach-Rufzeichen senden, auf Empfang schalten
Y	Auf Sendung schalten
Z	BU-Füllzeichen ausschalten
F2	Auf Empfang schalten, sobald FIFO leer, und Sprach-Rufzeichen senden (ohne CTRL)

Wirkung der Funktionstasten im Monitorprogramm:

F1	Kaltstart, RTTY-Empfang
F2	RX-Warmstart
F3	TX-Warmstart

Baudraten-Wahl (nach „V" bzw. „CTRL V"):

1	45 Bd (Amateur-Standard)
2	50 Bd (kommerzieller Standard)
3	75 Bd (noch selten)

Mark/Space-Wahl (nach „Q" bzw. „CTRL Q"):

1	1275 Hz (Space-Normalwert, 850 Hz Shift)
2	1675 Hz (Space für 450 Hz Shift)
3	1955 Hz (Space für 170 Hz Shift)
4	2125 Hz (Mark-Normalwert)

RX-Konverterpolarität (nach „R" bzw. „CTRL R"):

0	Mark = Low an PA 0
1	Mark = High an PA 0

TX-Zeilenformatierung (durchschnittliche Zeilenbreite, nach „W" bzw. „CTRL W"):

1	ca. 40 Zeichen
2	ca. 60 Zeichen

Literatur

[1] Anwendungsbeispiele für den Mikroprozessor 6502, Seite 43: Serielle ASCII-Eingabe per Interrupt. RPB 173, Fanzis-Verlag, München 1980.

[2] Ein RTTY-Baudot-Betriebsprogramm für den KIM-1. „ASCI" (Amateurfunk-Selbstbau-Computer-Informationsblatt) 1979, Heft 1 (auch in [1] abgedruckt).

[3] AIM-65 empfängt RTTY. FUNKSCHAU 1979, Heft 15, Seite 884.

[4] AIM-65 sendet RTTY. FUNKSCHAU 1979, Heft 16, Seite 953 und 1980, Heft 10, Seite 107.

[5] RTTY mit Sprachsynthese. FUNKSCHAU 1980, Heft 16, Seite 85.

[6] Sprachsynthese mit dem 6502. FUNKSCHAU-Sonderheft „Mikrocomputer-Anwendungen", 1980, Seite 71.

[7] RTTY mit dem AIM-65 – jetzt mit Drucker. FUNKSCHAU-Sonderheft „Programme für Kleincomputer und Taschenrechner", 1980, Seite 80.

[8] Amateurfunk-Sonderbetriebsarten: Funkfernschreiben. FUNKSCHAU 1980, Heft 23, Seite 85.

Stichworte zum Inhalt

Baudot-Code, RTTY, Mark, Space, 6502, FIFO, Interrupt-Timer, FSK-Konverter, Sprachsynthese.

Dr. Harald Lindner

Variablen listen mit dem CBM 3001

Welche Variablen wurden in einem Basic-Programm verwendet? Welche Werte haben sie nach Abbruch des Programms? Diese Fragen beantworten zwei Maschinensprache-Routinen, die wir hier für den CBM 3001 vorstellen.

Die beiden Programme Dump (*Bild 1*) und Array-Dump (*Bild 2*) sind in Maschinensprache geschrieben und erlauben den Ausdruck aller Variablen, die gerade im Speicher des CBM vorhanden sind, weil sie vorher durch ein Basic-Programm (nach RUN) definiert wurden. Die Werte von einfachen Variablen werden durch Starten von DUMP mit SYS845 auf den Schirm geschrieben, z. B.

AB = 1.2345678E-9
C1% = –888
Z$ = "BEISPIEL"
X2

Bei Fließkomma-, Integer- und Stringvariablen wird auch der aktuelle Wert ausgedruckt und kann mit Hilfe des Bildschirm-Editors, d. h. mit dem Cursor, geändert werden, indem man die alten Werte korrigiert und Return drückt. Die Variablen werden in der gleichen Reihenfolge auf den Schirm geschrieben, wie sie vorher im Basic-Programm definiert wurden.

Feldvariablen werden durch ARRAY-DUMP erfaßt. Dieses Programm wird mit SYS 845 gestartet und schreibt dann Feldvereinbarungen auf den Bildschirm, z. B.

DIMAB (1,2)
DIMEF% (3, 4, 5)
DIMH3$ (678)

Die Werte der Feldelemente selbst werden nicht ausgedruckt, da meist nur wenige Elemente wirklich interessieren

```
033A 20 3D 03 JSR 033D
033D B1 5C    LDA (5C),Y     ;Ein Zeichen des Variablennamens
033F 0A       ASL            ;Bit 7 fuer Variablentyp isolieren
0340 26 44    ROL 44         ;und zwischenspeichern
0342 4A       LSR            ;Bits 0-6 wiederherstellen
0343 99 42 00 STA 0042,Y     ;und speichern
0346 F0 03    BEQ 034B       ;fehlt 2-tes Zeichen? Dann -->
0348 20 D2 FF JSR FFD2       ;Akkumulator ausdrucken
034B C8       INY            ;Zeiger erhoehen
034C 60       RTS
034D 46 0D    LSR 0D         ;Ausgabeflag positiv machen
034F A5 2A    LDA 2A         ;Zeiger auf
0351 85 5C    STA 5C         ;Start der
0353 A5 2B    LDA 2B         ;Variablen
0355 85 5D    STA 5D         ;setzen
0357 38       SEC            ;Zeiger mit Anfang
0358 A5 5C    LDA 5C         ;der indizierten Variablen
035A E5 2C    SBC 2C         ;vergleichen
035C A5 5D    LDA 5D         ;Ist das Ende der
035E E5 2D    SBC 2D         ;einfachen Variablen erreicht?
0360 B0 EA    BCS 034C       ;Ja: Dann RTS
0362 A0 00    LDY #00        ;Index initialisieren
0364 84 44    STY 44         ;fuer Variablentyp
0366 20 3A 03 JSR 033A       ;Namen drucken und Typ merken
0369 A9 3D    LDA #3D        ;'='
036B A6 44    LDX 44         ;Variablentyp
036D F0 19    BEQ 0388       ;FLP? Dann -->
036F E0 02    CPX #02
0371 30 24    BMI 0397       ;String? Dann -->
0373 F0 41    BEQ 03B6       ;Funktion? Dann -->
0375 A2 25    LDX #25        ;'%' fuer Integer
0377 20 84 E7 JSR E784       ;X und A ausdrucken
037A B1 5C    LDA (5C),Y     ;Hi Byte Integer
037C 48       PHA            ;zwischenspeichern
037D C8       INY            ;Zeiger erhoehen
037E B1 5C    LDA (5C),Y     ;Lo Byte Integer
0380 A8       TAY
0381 68       PLA            ;Hi Byte wiederherstellen
0382 20 6D D2 JSR D26D       ;In Fliesspunktvariable verwandeln
0385 4C 91 03 JMP 0391       ;weiter wie bei Fliesspunkt
0388 20 D2 FF JSR FFD2       ;Akkumulator ausdrucken
038B 20 69 D0 JSR D069       ;Zeiger auf Variable
038E 20 AE DA JSR DAAE       ;Variable in FLP-Akku bringen
0391 20 E3 DC JSR DCE3       ;In String umwandeln und drucken
0394 4C B6 03 JMP 03B6
0397 A2 24    LDX #24        ;'$' fuer Stringvariable
0399 20 84 E7 JSR E784       ;X und A ausdrucken
039C A9 22    LDA #22        ;'"'
039E 20 D2 FF JSR FFD2       ;Akkumulator ausdrucken
03A1 B1 5C    LDA (5C),Y     ;Stringlaenge
03A3 AA       TAX            ;in X zwischenspeichern
03A4 C8       INY            ;Zeiger erhoehen
03A5 B1 5C    LDA (5C),Y     ;Lo Byte Stringadresse
03A7 85 1F    STA 1F         ;speichen fuer Druckroutine
03A9 C8       INY            ;Zeiger erhoehen
03AA B1 5C    LDA (5C),Y     ;Hi Byte
03AC 85 20    STA 20         ;ebenso speichen
03AE 20 23 CA JSR CA23       ;String drucken
03B1 A9 22    LDA #22        ;'"'
03B3 20 D2 FF JSR FFD2       ;Akkumulator ausdrucken
03B6 20 E2 C9 JSR C9E2       ;CRLF drucken
03B9 20 01 F3 JSR F301       ;Ist die STOP-Taste gedrueckt?
03BC F0 8E    BEQ 034C       ;Ja: Dann RTS
03BE A5 98    LDA 98         ;Ist die SHIFT-Taste gedrueckt?
03C0 D0 FC    BNE 03BE       ;Ja: Dann warten
03C2 18       CLC            ;Zeiger auf die
03C3 A5 5C    LDA 5C         ;naechste Variable
03C5 69 07    ADC #07        ;setzen
03C7 85 5C    STA 5C
03C9 A5 5D    LDA 5D
03CB 69 00    ADC #00
03CD 85 5D    STA 5D
03CF 4C 57 03 JMP 0357       ;Das Ganze nochmal
```

Bild 1. „DUMP" liefert alle Werte einfacher Variablen, soweit sie vorher von einem Basic-Programm definiert wurden

```
033A 20 3D 03 JSR 033D
033D B1 5C    LDA (5C),Y    ;Ein Zeichen des Variablennamens
033F 0A       ASL           ;Bit 7 fuer Variablentyp isolieren
0340 26 44    ROL 44        ;und zwischenspeichen
0342 4A       LSR           ;Bits 0-6 wiederherstellen
0343 99 42 00 STA 0042,Y    ;und speichern
0346 F0 03    BEQ 034B      ;fehlt 2-tes Zeichen? Dann -->
0348 20 D2 FF JSR FFD2      ;Akkumulator ausdrucken
034B C8       INY           ;Zeiger erhoehen
034C 60       RTS
034D 46 0D    LSR 0D        ;Ausgabeflag positiv machen
034F A5 2C    LDA 2C        ;Zeiger auf
0351 85 5C    STA 5C        ;Anfang der
0353 A5 2D    LDA 2D        ;indizierten Variablen
0355 85 5D    STA 5D        ;setzen
0357 38       SEC           ;Zeiger mit Ende
0358 A5 5C    LDA 5C        ;der Variablen
035A E5 2E    SBC 2E        ;vergleichen
035C A5 5D    LDA 5D        ;Ist das Ende
035E E5 2F    SBC 2F        ;der Variablen erreicht?
0360 B0 EA    BCS 034C      ;Ja: Dann RTS
0362 A2 44    LDX #44       ;'D'
0364 A9 49    LDA #49       ;'I'
0366 20 84 E7 JSR E784      ;X und A ausdrucken
0369 A9 4D    LDA #4D       ;'M'
036B 20 D2 FF JSR FFD2      ;Akkumulator ausdrucken
036E A0 00    LDY #00       ;Index initialisieren
0370 84 44    STY 44        ;fuer Variablentyp
0372 20 3A 03 JSR 033A      ;Namen drucken und Typ merken
0375 A9 24    LDA #24       ;'$' fuer Stringvariable
0377 A6 44    LDX 44        ;Variablentyp
0379 F0 09    BEQ 0384      ;FLP? Dann -->
037B E0 01    CPX #01
037D F0 02    BEQ 0381      ;String? Dann -->
037F A9 25    LDA #25       ;'%' fuer Integer
0381 20 D2 FF JSR FFD2      ;Akkumulator drucken
0384 A0 04    LDY #04
0386 B1 5C    LDA (5C),Y    ;Zahl der Indizes
0388 0A       ASL           ;mit 2 multiplizieren
0389 18       CLC           ;und
038A 69 04    ADC #04       ;4 addieren ergibt
038C A8       TAY           ;Offset fuer Adresse letzter Feldindex
038D A9 28    LDA #28       ;'('
038F 8D AA 03 STA 03AA
0392 A2 00    LDX #00       ;Setze fuer Druck des Feldindex
0394 86 07    STX 07        ;flag Variablentyp auf Integer
0396 B1 5C    LDA (5C),Y    ;Lo Byte Feldindex + 1
0398 38       SEC
0399 E9 01    SBC #01       ;1 abziehen
039B 85 60    STA 60        ;und speichern
039D 88       DEY           ;Y-Index vermindern
039E 98       TYA           ;und
039F 48       PHA           ;zwischenspeichern
03A0 B1 5C    LDA (5C),Y    ;Hi Byte Feldindex + 1
03A2 E9 00    SBC #00
03A4 85 5F    STA 5F        ;speichern
03A6 20 75 D2 JSR D275      ;In Fliesspunktvariable umwandeln
03A9 A9 28    LDA #28       ;zuerst '(', dann ',' (vgl. 038F/03B5)
03AB A0 01    LDY #01
03AD 20 F3 DC JSR DCF3      ;in String umwandeln
03B0 20 1C CA JSR CA1C      ;und String drucken
03B3 A9 2C    LDA #2C       ;','
03B5 8D AA 03 STA 03AA
03B8 68       PLA           ;Y-Index
03B9 A8       TAY           ;wiederherstellen
03BA 88       DEY           ;und dekrementieren
03BB C0 05    CPY #05       ;Ist der letzte Feldindex erreicht?
03BD 10 D3    BPL 0392      ;Nein: Dann naechster Feldindex
03BF A2 29    LDX #29       ;')'
03C1 A9 0D    LDA #0D       ;'CRLF'
03C3 20 84 E7 JSR E784      ;X und A ausdrucken
03C6 20 01 F3 JSR F301      ;Ist die STOP-Taste gedrueckt?
03C9 F0 81    BEQ 034C      ;Ja: Dann RTS
03CB A5 98    LDA 98        ;Ist die SHIFT-Taste gedrueckt?
03CD D0 FC    BNE 03CB      ;Ja: Dann warten
03CF A0 02    LDY #02       ;Offset fuer Adresse der Feldlaenge
03D1 18       CLC
03D2 B1 5C    LDA (5C),Y    ;Lo Byte Feldlaenge
03D4 65 5C    ADC 5C        ;Lo Byte Feldadresse
03D6 48       PHA           ;Lo Byte Adresse naechstes Feld
03D7 C8       INY           ;Offset fuer Hi Byte
03D8 B1 5C    LDA (5C),Y    ;Hi Byte Feldlaenge
03DA 65 5D    ADC 5D        ;Hi Byte Feldadresse
03DC 85 5D    STA 5D        ;Hi Byte Adresse naechstes Feld
03DE 68       PLA           ;Lo Byte wiederherstellen
03DF 85 5C    STA 5C        ;und speichern
03E1 4C 57 03 JMP 0357      ;Das Ganze nochmal
```

Bild 2. „ARRAY-DUMP" liefert die Dimensionierungs-Angaben aller von einem Basic-Programm definierten Feldvariablen

und die restlichen nur den Bildschirm füllen würden. Dazu würde man ohnehin besser ein kurzes Basic-Programm verwenden.

Beide Programme können mit der Shift-Taste unterbrochen werden. Läßt man die Shift-Taste wieder los, so wird das jeweilige Programm wieder fortgesetzt. Die Stop-Taste bricht beide Programme ab.

Statt auf den Bildschirm können die Variablenwerte auch auf Kassette, Drucker oder Floppy ausgegeben werden. Die Ausgabe auf Kassette erfolgt zum Beispiel mit folgenden Befehlen:

OPEN 6, 1, 1: CMD 6: SYS 845: PRINT # 6: CLOSE 6

Beide Programme passen einzeln in den zweiten Kassettenpuffer. Will man sie gleichzeitig im Speicher haben, so muß man eines davon in einen anderen Speicherbereich legen und die Sprungadressen entsprechend korrigieren. Die Routine 033A-034C ist für beide Programme gleich und braucht daher auch nur einmal im Speicher stehen.

CBM auf PET und rückwärts

Leider unterscheidet sich das „alte" Betriebssystem des PET-2001 ganz erheblich vom neueren CBM-3001-Betriebssystem. Das Umschreiben von Maschinensprache-Programmen wie dem vorstehenden oder auch von Basic-Programmen, die solche Befehle wie POKE, PEEK, SYS oder USR enthalten, erfordert daher das Wissen um verwendete Adressen in ROM und RAM dieser Computer.

Im FUNKSCHAU-Sonderheft „Mikrocomputer-Anwendungen" finden Sie ausführliche ROM- und RAM-Adressentabellen für die 6502-Computer KIM-1, AIM-65, PET-2001 und CBM-3001. Das Sonderheft ist bei vielen Zeitschriftenhändlern oder auch direkt beim Franzis-Verlag zu haben.

Berichtigung

Polarkoordinaten-Umrechnung in Basic

FUNKSCHAU 1980, Heft 26, Seite 92

In der Programmzeile 63 ist das X durch ein Y zu ersetzen.

Alfred Lenk

Mini-Bus

Wer seinen Minicomputer zu Meß- und Regelzwecken verwendet, den wird die begrenzte Anzahl von I/O-Leitungen schnell in seine Grenzen weisen. Abhilfe bringt der „Mini-Bus", der es erlaubt, einen 8-bit-Datenbus zu vervielfachen.

Wie so oft heißt auch hier das Zauberwort „Multiplex". Allerdings werden nicht einzelne Leitungen, sondern der gesamte Datenbus „gemultiplext". Dazu brauchen wir noch zwei zusätzliche Leitungen, um ein einfaches 2-Draht-Handshake-System aufzubauen. Wir verwenden somit ein Verfahren, das an den IEC-Bus erinnert, nicht ganz so viel Komfort hat, aber einfach aufzubauen ist, wobei sich der Bauteileaufwand in Grenzen hält.

Wie *Bild 1* zeigt, hängt das Interface am Datenbus und wird vom Datensender oder vom Datenbus selbst mit Daten belästigt. Mini-Bus hat nun die Funktion eines Schalters, der die Verbindung zwischen Daten-Sender/Empfänger und dem 8-bit-Datenbus herstellt, falls sich das Interface angesprochen fühlt. Dazu wartet Mini-Bus, bis die Adreß-Flag-Leitung (Bit 8) gesetzt ist, und interpretiert die Daten auf Bit 0...3 als Adresse, welche dann mit der eigenen, festeingestellten Adresse verglichen wird. Fällt dieser Vergleich positiv aus, fühlt sich das Interface angesprochen und schaltet den Daten-Sender/Empfänger auf den 8-bit-Bus durch.

Gleichzeitig antwortet das Interface auf der Rückmeldeleitung (Bit 9), und der Minicomputer weiß, daß sein Adressierungsversuch funktioniert hat.

Zusammen mit der Adresse kann dem Interface auf Bit 4...6 ein Datenwort mitgegeben werden, welches gespeichert wird und vielfältige Vorgänge im Daten-Sender/Empfänger auslösen kann (Meßbereichsumschaltung, Statusanzeige des Programms, Eingangswahlschalter usw.). Dieses 3-bit-Wort wird von allen Interfaces empfangen und ausgeführt – ob sie adressiert sind oder nicht! Weil als Adresse die Bits 0...3 verwendet werden, können 15 Interfaces parallel am 8-bit-Bus betrieben werden (die Adresse 0 darf auf keinen Fall verwendet werden!).

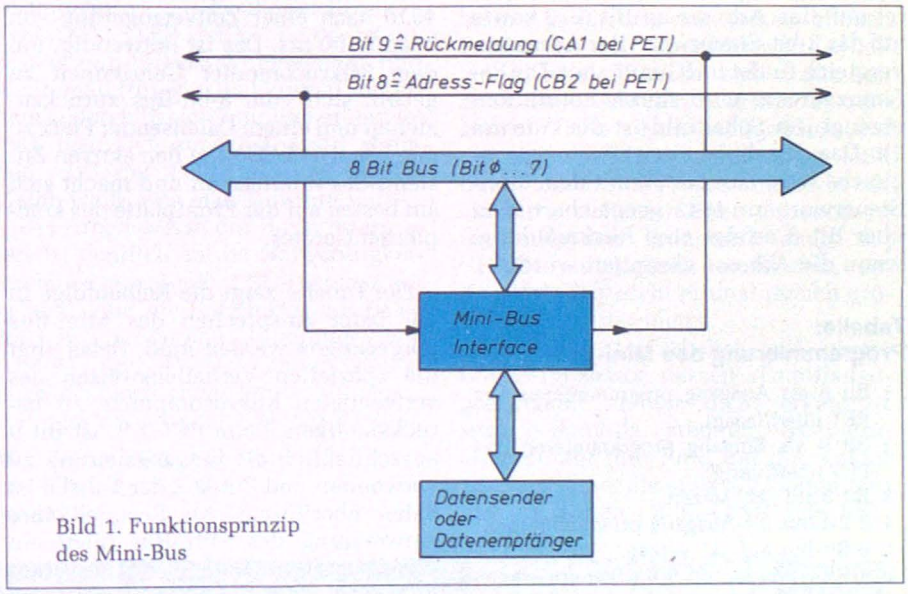


Bild 1. Funktionsprinzip des Mini-Bus

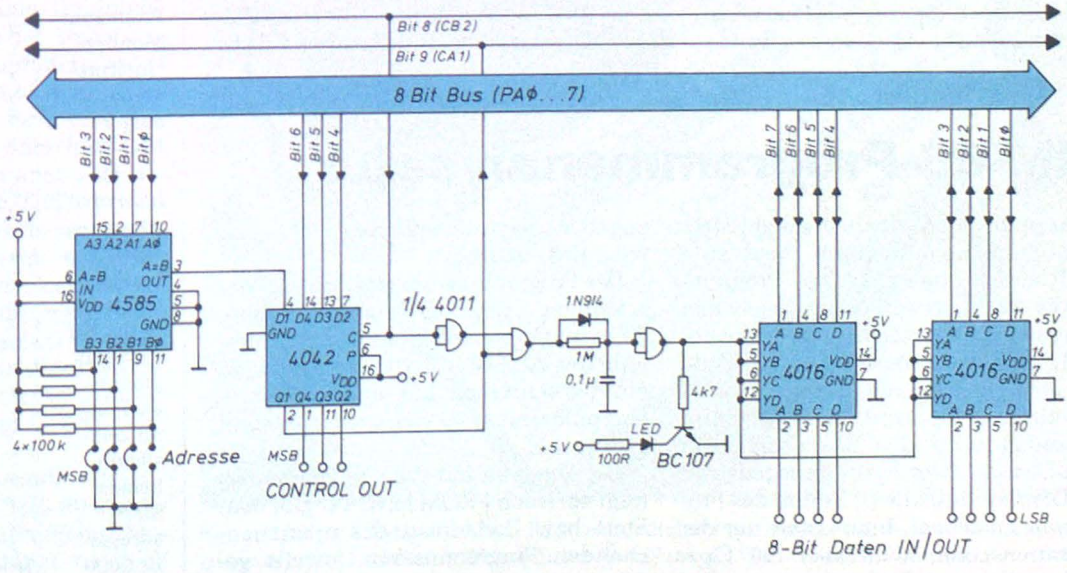


Bild 2. Die Gesamtschaltung des Mini-Bus läßt sich häufig noch mit dem Daten-Sender/Empfänger zusammen auf einer Europakarte aufbauen

```
10 INPUT"INTERFACE NR:":SC
11 INPUT"STEUERWORT":CC
20 GOSUB10000
30 PRINT"INTERFACE OK":GOTO10
10000 REM  ***BUS CONTROL***
10010 POKE59468,PEEK(59468)OR1:REM SET CA1 TRIGGER ^
10020 POKE59468,PEEK(59468)OR224:REM SET CB2 HI
10030 POKE59459,255:REM SET PORT TO OUTPUT
10040 POKE59471,0:REM INTERFACE CLEAR
10050 FC=PEEK(59457):REM CLEAR CA1
10060 POKE59471,SCOR16*CC:REM SET ADRESS
10070 POKE59468,(PEEK(59468)AND31)OR192:REM SET CB2 LO
10080 IFPEEK(59469)AND2THEN10100:REM CHECK CA1
10090 PRINT"FEHLER AUF SELECTCODE":SC:STOP
10100 POKE59459,0:RETURN:REM SET PORT INPUT
READY.
```

Bild 3. Beispiel einer Anwendung auf den PET

Mit fünf CMOS-Standardbauteilen kann die Schaltung aufgebaut werden (*Bild 2*). Wird die Bit-9-Leitung auf „H" gesetzt, ist das Interface passiv und wartet auf eine Adresse an Bit 0...3 sowie auf das 3-bit-Steuerwort. Der Adressenvergleich findet im IC 4585 statt. Die Referenzadresse wird durch Lötbrücken erzeugt (im Schaltbild ist die Adresse 1). Das Ergebnis eines Adressenvergleichs wird zusammen mit dem 3-bit-Steuerwort im 4042 gespeichert, und über Bit 8 erfolgt eine Rückmeldung, wenn die Adresse akzeptiert wurde.

Tabelle:
Programmierung des Mini-Bus

1. Bit 8 als Ausgang programmieren (bei PET überflüssig).
2. Bit 9 als Eingang programmieren (bei PET überflüssig).
3. Bit 8 auf „H" setzen.
4. 8-Bit-Bus als Ausgang programmieren.
5. 8-Bit-Bus auf „L" setzen.
6. 4-Bit Adresse und 3-Bit Steuerwort setzen.
7. Bit 8 auf „L" setzen.
8. Wenn hier Bit 9 nicht „H" ist, liegt ein Fehler vor.
9. 8-Bit-Bus als Eingang programmieren.
10. Rücksprung in das Hauptprogramm.

Wird die Bit-9-Leitung wieder auf „L" gesetzt und stimmen die Adressen überein, dann aktivieren die 3 NAND-Gatter die bidirektionalen Schalter 4016 nach einer Zeitverzögerung von knapp 100 ms. Das ist notwendig, um dem Mikrocomputer Gelegenheit zu geben, sich vom 8-bit-Bus zurückzuziehen und einem Datensender Platz zu machen. Die LED zeigt den aktiven Zustand des Interface an und macht sich am besten auf der Frontplatte des kompletten Gerätes.

Die *Tabelle* zeigt die Reihenfolge in der beim Ansprechen des Mini-Bus vorgegangen werden muß. Dabei sind die speziellen Verhaltensweisen des verwendeten Mikrocomputers zu berücksichtigen. Beim PET z.B. ist Bit 9 ausschließlich als Eingangsleitung zu verwenden und Punkt 2 der Tabelle ist daher überflüssig. Als Beispiel einer Anwendung des Mini-Bus dient ein PET-Programm (*Bild 3*), das unverändert auch auf CBM läuft. Beim Autor funktioniert das in FUNKSCHAU 16/80 beschriebene Digitalvoltmeter, durch Mini-Bus erweitert, mit Auto-Range und 16 Eingängen nebst diversen Relais-Registern am User-Port des CBM.

AIM-65-Programmanalysator

Das im *Bild* hexadezimal aufgelistete 6502-Maschinenprogramm dient zum Analysieren umfangreicher Programme: Es sucht in einem vorgegebenen Adressenbereich nach einem mnemonisch einzugebenden Operationscode und auf Wunsch auch nach dem einen bestimmten Operand. Bei Übereinstimmung wird die Befehlszeile im AIM-Disassembler-Format ausgedruckt (im Display sind alle (!) Zeilen des Programms zu sehen). Interessiert nur der Operationscode, nicht aber der Operandenteil, so ist für letzteren - - - einzugeben. Nach Beendigung der Suche wird END gedruckt.

Das Programm arbeitet nur bei eingeschalteter Drucker-Stromversorgung. Da es umfangreiche Monitorprogrammteile des AIM-65/PC-100 benutzt, wäre ein Umschreiben auf andere Mikrocomputer mit unvertretbarem Aufwand verbunden.

Der Umgang mit dem Programm erfolgt so: Nach FROM bzw. TO gibt man Start- bzw. Endadresse des zu untersuchenden Programms ein, jeweils gefolgt von Return. Hier einige Beispiele für das Absuchen von Speicherbereichen:

1. Wann wird ein bestimmtes Unterprogramm aufgerufen?
   OP-CODE: JSR
   OPERAND: 03A2 Return
   0251 20 JSR 03A2
   027B 20 JSR 03A2
   0285 20 JSR 03A2
   END
2. Auflisten aller absoluten Sprungbefehle
   OP-CODE: JMP
   OPERAND: Return
3. Zugriff auf eine bestimmte Operandadresse
   OP-CODE: - - -
   OPERAND: 035C Return
4. Zugriff auf Page 4 (d. h. 0400...04FF)
   OP-CODE:- - -
   OPERAND: 04- - Return
5. Welche Monitor-Unterprogramme werden aufgerufen?
   OP-CODE: JSR
   OPERAND: E- - - Return
   OP-CODE: JSR
   OPERAND: F - - - Return

Arno Hinrichs

```
                <M>=0000 20 13 EA 20
                < > 0004 A3 E7 20 D7
                < > 0008 E5 20 3B E8
                < > 000C 20 A7 E7 20
                < > 0010 13 EA A2 00
                < > 0014 20 82 00 A2
                < > 0018 00 A0 03 20
                < > 001C 72 00 20 90
                < > 0020 00 A2 09 20
                < > 0024 82 00 A2 04
                < > 0028 A0 07 20 72
                < > 002C 00 20 90 00
                < > 0030 A9 2D 85 F4
                < > 0034 38 AD 1C A4
                < > 0038 ED 25 A4 D0
                < > 003C 13 AD 1D A4
                < > 0040 ED 26 A4 D0
Wo wer-         < > 0044 0B A2 12 20
den in          < > 0048 82 00 20 90
einem           < > 004C 00 60 00 00
fremden         < > 0050 A9 01 8D 19
Programm        < > 0054 A4 20 2B E7
Monitor-        < > 0058 A6 F0 B5 F0
Unterpro-       < > 005C C9 2D F0 09
gramme          < > 0060 BD 68 A4 29
aufgeru-        < > 0064 7F D5 F0 D0
fen? Wo         < > 0068 06 CA D0 EE
werden          < > 006C 20 90 00 4C
I/O-Ports       < > 0070 34 00 20 73
angespro-       < > 0074 E9 C9 0D F0
chen? Das       < > 0078 06 E8 95 F0
und eini-       < > 007C 88 D0 F3 86
ges mehr        < > 0080 F0 60 B5 A0
findet der      < > 0084 C9 2A F0 07
AIM-Pro-        < > 0088 20 7A E9 E8
gramm-          < > 008C 4C 82 00 60
analysa-        < > 0090 A9 80 8D 16
tor selb-       < > 0094 A4 8D 11 A4
ständig         < > 0098 20 24 EA 0E
heraus.         < > 009C 11 A4 60 00
Start-          < > 00A0 4F 50 2D 43
adresse         < > 00A4 4F 44 45 3A
ist 0000        < > 00A8 2A 4F 50 45
                < > 00AC 52 41 4E 44
                < > 00B0 3A 2A 20 45
                < > 00B4 4E 44 2A 00
                <M>=0112 4C 00 00 04
```

Basic in der hohlen Hand

Der bisher wohl kleinste in der Programmiersprache Basic bedienbare Rechner wird von Sharp hergestellt und u. a. vom Großversandhaus Quelle sowie von Tandy vertrieben. Lesen Sie, was von diesem Gerät zu halten ist.

Der Taschencomputer PC-1211 von Sharp ähnelt äußerlich eher einem etwas breit geratenen Taschenrechner als einem Basic-Computer, zumindest, was seine Größe angeht: 17,5 cm breit, 7 cm hoch, rund 1,5 cm dick. Sieht man etwas genauer hin, so bemerkt man gleich, daß das Ding doch nicht nur ein gewöhnlicher Rechner sein kann: Unter dem 24stelligen LCD-Punktmatrixdisplay befindet sich eine vollständige alphanumerische Tastatur in schreibmaschinenähnlicher QWERTY-Anordnung mit getrenntem Nummernfeld. Mit einer MODE-Taste kann man das Gerät durch mehrere Betriebsarten durchtasten: DEF (Aufruf von mit Namen versehenen Programmen); RUN (Ausführen von Basic-Programmen und Berechnung arithmetischer Ausdrücke); PRO (Eingabe von Basic-Programmen) und schließlich RESERVE, wobei kurze Programme mit Namen versehen werden können.

Außerdem kann man auch in der PRO-Betriebsart in bestimmte Zeilen des Basic-Programms Namen in Anführungszeichen schreiben. Hat man dann mehrere Basic-Programme im (nichtflüchtigen) Arbeitsspeicher stehen, so z. B. eines mit Zeilennummern ab 10 und eines ab 200, so kann man diese zwar wie bei anderen Basic-Interpretern auch mit RUN 10 oder RUN 200 starten – hier geht's aber auch z. B. mit RUN"SORT" bzw. RUN"TEST", wenn man diese Namen als Labels verwendete. Das funktioniert auch mit LIST: Hier darf statt der Angabe einer Zeilennummer auch ein in Anführungszeichen stehender Name angegeben werden. Darüber hinaus kann man diese symbolische Zeilenangabe auch bei Sprungbefehlen wie GOTO oder GOSUB verwenden.

Überhaupt ist das verwendete Basic – es ähnelt etwas dem des TRS-80 und ist in rund 7 KByte ROM untergebracht – recht komfortabel, gestattet es doch Stringverarbeitung (Variablen mit Texten usw.) und besitzt solche nicht selbstverständlichen Befehle wie PRINT USING (Anzeige von Zahlen im gerundeten Festkommaformat), FOR...TO...STEP...NEXT. Leider ist die Syntax zum Teil etwas ungewohnt, und wenn man früher schon einmal mit einem anderen Basic-Computer arbeitete, muß man sich zunächst damit abfinden, daß der PC-1211 nicht immer gleich weiß, was man von ihm will.

Die Arbeitsweise des Basic-Interpreters ist deutlich an die Notwendigkeiten eines einzeiligen LCD-Displays angelehnt, und die Entwickler des Basic solcher „größeren" Computer wie AIM-65 und PC-100 hätten davon durchaus noch etwas lernen können.

Programmiert man z. B. die Schleife

```
10 "TEST" FOR A=1 TO 5
20 PRINT A: NEXT A
30 END
```

so würde auf dem Display eines AIM-65 praktisch nichts als ein bißchen Flakkern und dann die Zahl 5 zu sehen sein, da dieses Programm schlicht und einfach die Zahlen 1...5 ausgibt. Beim PC-1211 sieht man nach RUN"TEST" zunächst eine 1 im Display. Erst nach Druck auf „Enter" erscheint die nächste Zahl. Will man dagegen absichtlich vermeiden, daß das Programm bei jeder Ausgabe anhält, so kann man PRINT durch PAUSE ersetzen, so daß der jeweilige Wert eine runde Sekunde im Display erscheint, bevor das Programm weiterläuft. Einen Haken hat die Sache allerdings: Nach PRINT oder PAUSE darf nicht ein beliebiger arithmetischer Ausdruck stehen, und man sollte lieber nur einen gewöhnlichen Variablennamen verwenden.

Obwohl der Speicher des Taschencomputers nicht flüchtig ist – auch beim Ausschalten des Gerätes bleiben Programme und Variablenwerte erhalten –, ist es nützlich, sich zusätzlich noch das Kassetten-Interface zuzulegen, ein flaches Kunststoffgehäuse, auf das sich der PC-1211 aufstecken läßt. Damit lassen sich Programme nicht nur einzeln speichern und abrufen, sondern im laufenden Programm auch mit CHAIN automatisch nachladen, ein Komfort, den nicht einmal manch größerer Computer bietet.

Der PC-1211 arbeitet mit einem CMOS-Prozessor, dessen „Familienzugehörigkeit" leider nicht erkennbar war; jedenfalls arbeiten Basic-Programme auf ihm rund um den Faktor 100 langsamer als etwa auf Computern mit „richtigen" 8-bit-CPUs wie PET,

Rund 11 KByte ROM und 1 KByte RAM beinhaltet der von Sharp hergestellte Basic-Taschencomputer PC-1211, der von Tandy unter dem Namen „TRS-80 Pocket Computer" vertrieben wird

AIM-65 usw. Allerdings dürfte das bei der Mehrzahl der Anwendungen kaum eine Rolle spielen, wenngleich diese Tatsache vermuten läßt, daß der Hardware-Aufbau des PC-1211 eher der Struktur eines programmierbaren Taschenrechners denn der eines echten Mikrocomputers mit seinen Bus-Systemen ähnelt.

Sicher ist der Basic-Taschencomputer PC-1211 also kein Gerät, das nun alle anderen Basic-Computer überflüssig macht. Das einzeilige LCD-Display, die geringe Arbeitsgeschwindigkeit, die ziemlich winzige Tastatur, das Fehlen einer Schnittstelle für einen Drucker, aber auch die vergleichsweise kleine Arbeitsspeicher-Kapazität (rund 1 KByte) bedeuten durchaus wesentliche Einschränkungen. Für viele Anwender dürfte das Gerät jedoch eine höchst interessante Alternative zu den programmierbaren Taschenrechnern der neueren Generation darstellen: Ein Basic-Programm ist nun einmal wesentlich übersichtlicher und auch leichter änderbar als eine Folge von merkwürdigen Tastencodes, deren Bedeutung man erst im Rechner-Handbuch nachsehen muß. Sharp tat also einen Schritt in die richtige Richtung – warten wir ab, was weiter passiert. Fe.

„Screen Editing" mit dem PC-100

Für den mit dem AIM-65 baugleichen Mikrocomputer PC-100, der im Grundzustand über eine 20stellige alphanumerische LED-Anzeige und einen Thermodrucker verfügt, vertreibt der Hersteller Siemens nun ein Video-Interface, das im Vergleich zu bisher erhältlichen Versionen recht komfortabel ist.

Das Interface kostet rund 700 DM und wird direkt an den Adressen- und Datenbus des PC-100 angeschlossen. Alle Steuer- und Formatieraufgaben erfolgen per Software, wobei das hierfür notwendige Betriebsprogramm innerhalb des Interface-Gehäuses in einem EPROM 2716 untergebracht ist. Ein weiteres EPROM, ein 2758, dient als Zeichengenerator für Groß- und Kleinbuchstaben, alle ASCII-Sonderzeichen und einige einfache Grafiksymbole.

Das Besondere an diesem Video-Interface ist wohl, daß – ähnlich wie bei PET und CBM – das Editieren von Texten und Programmen auf dem Bildschirm möglich wird (Screen Editing): Man kann mit dem Cursor auf dem Bildschirm umherfahren, Korrekturen ausführen und dann die geänderte Programmversion in den Arbeitsspeicher übernehmen. Für das Editieren stehen auch Insert bzw. Delete von einzelnen Zeichen oder Zeilen zur Verfügung.

Das PC-100-Video-Interface liefert ein Bildschirmformat von ausnützbaren 28 Zeilen mit je 64 Zeichen. Zusätzlich erscheint unter einem horizontalen Trennungsstrich im unteren Teil des Bildschirms eine „Statuszeile", in der die Betriebsart (z. B. INS für den Insert-Modus) und die aktuelle Cursorposition angezeigt werden (Zeile/Spalte). Die hohe Zeichenzahl erfordert einen guten Monitor; bewährt hat sich z. B. der grünleuchtende Sanyo-Datenmonitor DM5912.

Als Video-RAM enthält das Interface vier 2114-Chips (2 KByte, Adressenbereich 9000...97FF). Zusätzlich ist noch eine RAM-Page (2 x 2112) eingebaut, die bei 9F00...9FFF liegt und vom EPROM-Betriebsprogramm (9800 bis 9EFF) nur von 9F00...9F0E belegt wird; der Rest steht dem Anwender zur Verfügung. Schaltet man (mit CTRL C) das Video-Interface aus, so läßt sich das 2-K-Video-RAM auch als ganz normaler Arbeitsspeicher verwenden, z. B. für die Symboltabelle des Assemblers. Die auf einer Europakarte untergebrachte Video-Aufbereitung besteht im wesentlichen aus 25 TTL-ICs in Low-Power-Schottky-Ausführung, so daß die Stromversorgung noch aus dem PC-100-Netzteil erfolgen kann.

Natürlich hat das Interface aber auch seine „Macken". So wird z. B. der Timer 2 des User-VIA verwendet, was leicht zu Konflikten mit früher geschriebenen Anwenderprogrammen führt, die ebenfalls diesen Timer verwenden.

Ähnlich konfliktträchtig sind Anwenderprogramme, die CTRL-Tastenkombinationen und -zeichen verwenden; hier wird man beim Einsatz des Video-Interface nicht umhin kommen, solche Programme gründlich zu überarbeiten. Fe.

Berichtigung

TRS-80-Disassembler

FUNKSCHAU 1980, Heft 13, Seite 89

Beim Disassemblieren der Operationscodes DD09, DD19, DD39, FD09 und FD19 tritt fälschlicherweise eine Fehlermeldung auf – ein falscher Fehler also! Dies kann durch folgende Änderung beseitigt werden:

6032:AF 32 0B 65 00 00 00

Beim Übersetzen der Befehle ED5F und ED4F wird eine Fehlermeldung ausgegeben, da diese Befehle nur für Testzwecke bestimmt sind und nicht in Programmen verwendet werden sollen.
Hubert Holewa

µC-Nachrichten

Unter dem Namen **Computershop GmbH** haben sich kürzlich R. Nedela (bisher: Microshop Bodensee) und F. Monninger (bisher: Computershop Eschborn) zusammengeschlossen und die bisherigen Firmen aufgelöst. Hauptsitz des Computershops ist 7778 Markdorf; 6236 Eschborn dient als Filiale. Verkauft werden Systeme wie AIM-65, SYKO 100, DAI, Sorcerer und S-100-Systeme.

F'

Erste Erfahrungen mit den neuen **16-bit-Mikroprozessoren** sind ziemlich ernüchternd: So stellte sich heraus, daß Motorolas 68000 in der derzeit erhältlichen 4-MHz-Version bei typischen Benchmark-Tests langsamer arbeitet als die 8-bit-CPU 6502 bei 1-MHz-Taktfrequenz. 16-bit-Geschwindigkeitsvorteile sind also nicht unbedingt automatisch zu erreichen.

Basic-Texteditor

In FUNKSCHAU 1980, Heft 22, fand sich auf Seite 91 eine verbesserte Version des ursprünglich in Heft 1/1980 veröffentlichten Texteditors. Dabei dienten die Programmzeilen 980 und 985 der Vertauschung von Groß- und Kleinbuchstaben zur Anpassung des Rechner-Zeichensatzes an einen Centronics-Drucker. Diese beiden Zeilen können unter Ausnutzung einer im Sonderheft „Programme für Kleincomputer und Taschenrechner" genannten Alternative zu IF-THEN-Anweisungen durch eine einzige ersetzt werden:

980 AS = ASC (B$): B$ = CHR$(AS+(AS > 128 OR AS < 64) *128*SGN (AS-128))

Dabei wird ausgenützt, daß Vergleichsoperationen einen direkt verwendbaren Zahlenwert, nämlich 0 oder –1, liefern. Fa. Gerb GmbH

Hans-Georg Joepgen

Großputz hilft bei der Dateipflege

Das Disketten-Betriebsprogramm DISK II (Versionen DOS 3.2 und DOS 3.21) zählt zwar zum Besten, das gegenwärtig für kleine Rechnersysteme angeboten wird, doch läßt selbst dieses für die Rechner der Apple-Klasse einschließlich des Mikrocomputers ITT 2020 bestimmte Programm noch vielerlei Wünsche offen. Der folgende Beitrag zeigt, wie man mit Hilfe eines Basic-Programmes mäßiger Länge dem Disketten-Betriebssystem hilfreich unter die Arme greifen kann. Darüber hinaus werden zwei Programmiertechniken erläutert, die auch für die Arbeit mit anderen Microsoft-Rechnern wie zum Beispiel dem TRS-80 nützliche Anregung sein können: Arbeiten mit geschützten Feldern auf dem Schirm und selektive Fehlerbehandlung.

Das Löschen von Diskettenaufzeichnungen beliebiger Art geschieht im System DISK II durch das Aussenden des Steuerzeichens Control-D(Hexwert 04), dem der in ASCII-Zeichen abgebildete Befehl „UNLOCK" sowie, ebenfalls als ASCII-Kette, der Name der Aufzeichnung folgt. Hierdurch wird die Aufzeichnung zum Löschen freigegeben. Danach muß der Vorgang wiederholt werden, wobei die Kette „UNLOCK" durch die Zeichen „DELETE" zu ersetzen sind; nur für ungesicherte Files ist die UNLOCK-Prozedur nicht erforderlich. All dies geschieht, sowohl beim Aufruf durch ein Programm als auch durch unmittelbares Tastatur-Kommando, mit ausreichender Geschwindigkeit – solange nur eine einzige Aufzeichnung zu löschen ist. Will man dagegen, etwa bei Pflegearbeiten an größeren Dateien, gleich ein paar Dutzend Aufzeichnungen selektiert entfernen, dann artet die Standard-Löschprozedur der Apple-Maschinen schnell in Sklavenarbeit aus. In solchen Fällen wünscht man eine für derartige Großputz-Unternehmen besser geeignete Handreichung, die auch „fehlerfest" sein sollte: Das heißt: Ein Programm, das auf Bedienungsfehler wie falsches Eintippen von Aufzeichnungs-Namen nicht gleich mit Abbruch antwortet. Diesen Anforderungen kommt „Blocklöschung" nach und bietet darüber hinaus einiges an zusätzlichem Komfort.

String-Matrix als Zwischenspeicher

Auf eine ausführliche Besprechung von „Blocklöschung" soll hier verzichtet werden, da das Programmlisting selbst (*Bild 1*), die Variablen-Referenztabelle (*Bild 2*) und ein Katalog der Sprungziele mit Kommentaren (*Bild 3*) gegebenenfalls Einzelfragen erschöpfend beantworten. Statt dessen seien die dem Programm zugrundeliegenden Hauptalgorithmen hier kurz erläutert: Das Programm fordert zur Namenseingabe auf (bis zu 32 Files möglich), bietet nach Eingabeabschluß durch einen Leerstring ein Verzeichnis der zwischengespeicherten Namen an und läßt die Wahl, ob der Operator den Inhalt des Verzeichnisses akzeptiert, oder ob Änderungen gewünscht werden. Ist dies der Fall, so kann dem Ersatz eventuell fälschlich eingetippter Bezeichnungen durch eine korrekte Fassung oder dem ersatzlosen Streichen beliebig vieler File-Namen gewählt werden. Gelingen einzelne Löschoperationen nicht, weil sich eine entsprechend benannte Aufzeichnung beispielsweise nicht auf der Scheibe befand, so führt das Programm über derartige fehlgeschlagene Versuche Buch und meldet die hiervon betroffenen Namen bei Programm-Schluß. Programmtechnisch realisiert wurde „Blocklöschung" durch die Errichtung einer eindimensionalen Zeichenketten-Matrix mit 33 Elementen, von denen die 32 obersten genutzt sind. Auf diese String-Matrix wird mit Hilfe eines Index (Variablen-Namen N) zugegriffen.

„Fenster" schaffen Ordnung auf dem Schirm

In der mittleren Datentechnik hat sich seit Jahren eine Art der Schirmausgabe bewährt, die – je nach Hardware-Hersteller – als „Formularbetrieb" oder „Ausgabe mit geschützten Schirmfeldern" bezeichnet wird. Grundgedanke: Das Betriebssystem des Rechners wird so modifiziert, daß dem Beutzerprogramm nicht länger der gesamte Schirm für Ausgaben zur Verfügung steht, sondern nur noch der Bereich außerhalb von „geschützten Feldern". Das Programm „Blocklöschung" nimmt nun eine Dreiteilung des Schirmes vor. Im oberen Bilddrittel erfolgen wechselnde Mitteilungen an den Benutzer, im unteren Schirmteil werden die jeweils letzten zehn Aufzeichnungs-Namen dargestellt. Außerhalb dieser beiden „Fenster" sind die während des Programmablaufs unverändert bleibenden Texte angeordnet, zu denen beispielsweise der Tabellenkopf über der Namens-Liste gehört. Realisiert ist der Bereichsschutz durch die Manipulation zweier Speicherstellen auf der Seite 0 des RAM-Bereichs, in denen die Rechner der Apple-Klasse ihre Werte für den Zeilenstart („Top of Screen", dezimal 34) und die unterste zur Verfügung stehende Zeile („Bottom of Screen", dezimal 35) festhalten. Durch Veränderungen dieser Pointer, wie sie von den Subroutinen „Fenster oben", „Fenster unten" und „Fenster aufheben" vorgeführt werden, gelangt man bequem zu einer Betriebsart „Field protect", wie sie sonst nur von wesentlich teureren und größeren Rechnersystemen zur Verfügung gestellt wird.

„Blocklöschung" erkennt Fehler

Damit vermieden wird, daß sich das Programm bereits nach einem einzigen vergeblichen Suchvorgang verabschiedet, wiewohl noch viele korrekt geschriebene Aufzeichnungs-Namen in der Stringmatrix einer Behandlung entgegensehen, wurde in Zeile 75 das Statement ONERR GOTO 96 eingefügt. Die Anweisung (in anderen Basic-Dialekten auch TRAP geschrieben) setzt

```
 1  REM ************************
 2  REM *                      *
 3  REM *   BLOCKLOESCHUNG     *
 4  REM *   --------------     *
 5  REM *                      *
 6  REM * HG JOEPGEN/17.8.80  *
 7  REM *                      *
 8  REM ************************
 9  REM
10   REM TITELTAFEL
11   REM ----------
12 D$ =  CHR$ (4): DIM A$(64): HOME
13   PRINT "BLOCKLOESCHUNG"
14   PRINT "==============": PRINT " "
15   PRINT "GEBEN SIE DIE NAMEN DER ZU"
16   PRINT "LOESCHENDEN FILES EIN."
17   PRINT "JEWEILS GEFOLGT VON <RETURN>.-"
18   PRINT " "
19   PRINT "ABSCHLUSS: LEERSTRING.-"
20   REM
21   REM NAMEN-EINGABE
22   REM -------------
23   PRINT " "
24   PRINT "NR  FILE-NAMEN"
25   FOR I = 0 TO 39: PRINT "-";: NEXT : PRINT "-"
26   GOSUB 114: VTAB 12:N = 0
27 N = N + 1: IF N > 32 THEN 67
28   GOSUB 105
29   INPUT " ";A$
30   IF A$ < > "" THEN A$(N) = A$: GOTO 27
31 NMAX = N - 1:N = 0: HOME
32   REM
33   REM KONTROLL-LISTE
34   REM --------------
35 N = N + 1: GOSUB 105: PRINT " "A$(N)
36   IF (N < NMAX) AND (N / 10 <  >  INT (N / 10)) THEN 35
37   GOSUB 110
38   PRINT "KONTROLLIEREN SIE DIE FILE-"
39   PRINT "NAMEN BITTE."
40   VTAB 7: HTAB 3: PRINT " +++FREI ZUR LOESCHUNG (J/N) ? ";
41   GET A$: IF (A$ <  > "J") AND (A$ <  > "N") THEN  PRINT  CHR$ (7): GOTO
      40
42   IF A$ = "N" THEN 48
43   IF N = NMAX THEN 73
44   GOSUB 114: GOTO 35
45   REM
46   REM KORREKTUR
47   REM ---------
48   HOME : PRINT " FILE-NAMEN  AENDERN (A)": PRINT " "
49   PRINT "       ODER ENTFERNEN (E)? ";
50   GET A$: IF (A$ <  > "A") AND (A$ <  > "E") THEN  PRINT  CHR$ (7): GOTO
      48
51   IF A$ = "A" THEN 61
52   REM
53   REM NAMEN ENTFERNEN
54   REM ---------------
55   HOME : INPUT "NUMMER DES ZU ENTFERNENDEN NAMENS? ";I
56   IF (I < 1) OR (I > NMAX) OR (I <  >  INT (I)) THEN  PRINT  CHR$ (7): GOTO
      55
57 NMAX = NMAX - 1: FOR N = I TO NMAX:A$(N) = A$(N + 1): NEXT : GOSUB 114:
     N = 0: GOTO 35
58   REM
59   REM NAMEN AENDERN
60   REM -------------
61   HOME : INPUT "NUMMER DES ZU AENDERNDEN NAMENS? ";I
62   IF (I < 1) OR (I > NMAX) OR (I <  >  INT (I)) THEN  PRINT  CHR$ (7): GOTO
      61
63   HOME : PRINT "NEUER NAME BITTE:": PRINT " ": INPUT " ";A$(I): GOSUB 11
      4:N = 0: GOTO 35
64   REM
65   REM ARRAY VOLL
66   REM ----------
67   VTAB 23: PRINT " +++WEITERE EINGABEN NICHT MOEGLICH."
68   PRINT "      MIT BEL.  TASTE QUITTIEREN."
69   GET A$: GOTO 31
70   REM
71   REM LOESCHEN
72   REM --------
73   PRINT :N = 0
74 N = N + 1: IF N > NMAX THEN 81
75   ONERR  GOTO 96
76   PRINT D$"UNLOCK"A$(N)
77   PRINT D$"DELETE"A$(N):A$(N) = "": GOTO 74
78   REM
79   REM REGULAER-ENDE
80   REM -------------
81   IF  NOT FEHLERFLAG THEN  GOSUB 118: VTAB 24: PRINT " +++ PROGRAMM FEHL
      ERFREI AUSGEFUEHRT.-";: END
82   REM
83   REM KORRIGIERBARE FEHLER
84   REM --------------------
85   HOME : PRINT " +++ PROGRAMM BEENDET.": PRINT " "
86   PRINT "      NICHT ANGETROFFEN:"
87   GOSUB 114:N = 0:I = 0
88 N = N + 1: IF N > NMAX THEN  GOSUB 118: END
89   IF A$(N) = "" THEN 88
90 I = I + 1: IF (I / 10 =  INT (I / 10)) THEN  VTAB 24: PRINT " +++ MIT B
      ELIEBIGER TASTE QUITTIEREN";: GET A$: HOME
91   GOSUB 105: PRINT " "A$(N)
92   GOTO 88
93   REM
94   REM FEHLERBEHANDLUNG
95   REM ----------------
96   IF  PEEK (222) = 6 THEN FEHLERFLAG = 1: GOTO 74
97   REM
98   REM UNKORRIGIERBARER FEHLER
99   REM -----------------------
100   GOSUB 118: PRINT "+ ABBRUCH NACH UNKORRIGIERBAREM FEHLER.-"
101   END
102   REM
103   REM ZAEHLERSTAND DRUCKEN
104   REM --------------------
105 N$ =  STR$ (N): IF N < 10 THEN N$ = " " + N$
106   PRINT N$;: RETURN
107   REM
108   REM FENSTER OBEN
109   REM ------------
110   POKE 34,3: POKE 35,9: HOME : RETURN
111   REM
112   REM FENSTER UNTEN
113   REM -------------
114   POKE 34,11: POKE 35,24: HOME : RETURN
115   REM
116   REM FENSTER AUFHEBEN
117   REM ----------------
118   POKE 34,0: POKE 35,24: RETURN
```

Bild 1. Das Programm „Blocklöschung“ dient zum selektiven Entfernen von Dateien aus Floppy-Aufzeichnungen bei Apple-II und ITT-2020

```
    ********************************
    *                              *
    *       BLOCKLOESCHUNG         *
    *                              *
    *  -->TABLE OF VARIABLES<--    *
    *                              *
    ********************************

A$ - ZEICHENKETTEN-AKKU
29 30 30 41 41 41 42 50 50
50 51 69 90

A$(*) - ZEICHENKETTEN-ARRAY
12 30 35 57 57 63 76 77 77
89 91

D$ - DOS-STEUERZEICHEN
12 76 77

FEHLERFLAG - GESETZT NACH "F. NOT FOUND"
81 96

I - INKREMENTALINDEX
25 55 56 56 56 56 57 61 62
62 62 62 63 87 90 90 90 90

N - EINGABEZEAHLER, ZWEITINDEX
26 27 27 27 30 31 31 35 35
35 36 36 36 43 57 57 57 57
63 73 74 74 74 76 77 77 87
88 88 88 89 91 105 105

N$ - FORMATIERTE ORDNUNGSKETTE
105 105 105 106

NMAX - INDEX FUER LETZTE EINGABE
31 36 43 56 57 57 57 62 74
88

END OF VAR. LIST
```

Bild 2. Die mit „Appledoc" gefertigte Referenzliste für Variable erleichtert eine Analyse von „Blocklöschung"

ein systeminternes Flag, das Programm-Unterbrechungen beim Auftreten einer Fehlerbedingung unterbindet. Stattdessen wird hier ein unbedingter Sprung nach der Basic-Zeile 96 ausgeführt. In der Speicherstelle dezimal 222 führen Monitor und DISK II gemeinsam Buch über die Ursache eines Fehlers. Nur dann, wenn das Disk Operating System „File nicht gefunden" meldet, findet sich in 222 der Wert 6. Davon macht Zeile 96 Gebrauch, um eine Basic-Variable FEHLERFLAG zu setzen und säuberlich zwischen solchen Fehlern zu unterscheiden, die das Programm aus eigener Kraft zu meistern vermag, und solchen, auf die es weiterhin mit Abbruch reagieren sollte.

Benutzer des soeben auf dem Markt erschienenen Disketten-Betriebssystems DOS 3.3 benötigen „Großputz" nicht; ihnen bietet das im Lieferumfang enthaltene File-Entwicklungs-Paket FID verwandte Handreichungen.

Literatur

[1] *Valentini, Johannes:* „Ein (fast) ideales Speichermedium – das Disk Operating System des Euro-apple." – FUNKSCHAU 1980, Heft 6, S. 105...107. Franzis, München.

[2] *Joepgen, Hans-Georg:* „Der Euro-Apple". FUNKSCHAU 1979, S. 831...834. Franzis, München.

[3] *Joepgen, Hans-Georg:* „Variablen-Wächter sorgt für mehr Programm-Transparenz." ELEKTRONIK 1980, Heft 8, S. 87...88. Franzis, München.


```
    ********************************
    *                              *
    *       BLOCKLOESCHUNG         *
    *                              *
    *  -->TABLE OF LN# XREFS<--    *
    *                              *
    ********************************

    27 - EINGABEZAEHLER INKREMENT.
30

    31 - INIT FUER KONTROLL-LISTE
69

    35 - KONTROLL-LISTE DRUCKEN
36 44 57 63

    40 - BENUTZER-WUNSCH ERFRAGEN
41

    48 - AENDERUNGSART ERFRAGEN
42 50

    55 - NAMEN-NUMMER ERFRAGEN
56

    61 - N-NUMMER AENDERUNG ERFR.
51 62

    67 - "ARRAY VOLL" MELDEN
27

    73 - INIT LOESCHROUTINE
43

    74 - AUF ENDE PRUEFEN
77 96

    81 - AUF FEHLERFLAG PRUEFEN
74

    88 - ENDE DER FEHLERLLISTE
89 92

    96 - DOS-FEHLERREGISTER PRUE.
75

   105 - N FORMATIERT AUSDRUCKEN
28 35 91

   110 - FENSTERPOINTER AUF "OBEN"
37

   114 - FENSTERPOINTER AUF "UNTEN"
26 44 57 63 87

   118 - AUF "VOLLEN SCHIRM" GEHEN
81 88 100

END OF LN# LIST
```

Bild 3. Ebenfalls mit „Appledoc" entstand diese Sprungziel-Referenztabelle

Textrettungs-Programm für den AIM-65

Mit diesem Programm kann man Sourcedateien in den Editor-Bereich einlesen, die mit der AIM-Software nicht mehr zu laden sind. Das AIM-Betriebssystem lehnt Dateien ohne Block 00 und Kennsatz, mit Blockfolge-Fehler oder mit Bitsummenfehler einfach ab. Auf die Gefahr hin, daß einige Daten fehlen oder falsch sind, wird die Datei geladen. Anhand der Auflistung kann

```
                <M>=0000 00 00 00 53
                < > 0004 4F 55 52 43
                < > 0008 45 2D 43 4F
                < > 000C 44 45 20 52
                < > 0010 45 54 54 45
                < > 0014 4E A2 00 20
                < > 0018 44 EB 20 3B
                < > 001C E8 B5 03 20
                < > 0020 05 EF E8 E0
                < > 0024 12 90 F6 D8
                < > 0028 20 EA ED 20
                < > 002C 29 EE C9 23
                < > 0030 F0 06 C9 16
                < > 0034 D0 F2 F0 F3
                < > 0038 20 29 EE 85
                < > 003C 02 A9 00 85
                < > 0040 01 20 29 EE
                < > 0044 A0 00 91 E1
                < > 0048 C9 0D D0 0F
                < > 004C A6 00 E0 0D
                < > 0050 D0 03 4C A1
                < > 0054 E1 A2 0D 86
                < > 0058 00 D0 04 A2
                < > 005C 00 86 00 18
                < > 0060 A9 01 65 E1
                < > 0064 85 E1 A9 00
                < > 0068 65 E2 85 E2
                < > 006C A5 01 18 69
                < > 0070 01 85 01 C9
                < > 0074 4F 90 CA 20
                < > 0078 44 EB 20 3B
                < > 007C E8 A5 02 20
                < > 0080 46 EA 4C 28
                < > 0084 00 02 85 E4
                <K>↵=10C
                /
                 010C 4C JMP 0015
```

In schwierigen Fällen hilft dieses Programm, Texte von der Kassette zu laden – auf die Gefahr hin, daß Fehler nicht erkannt werden

das Programm leicht ergänzt oder berichtigt werden.

Zunächst wird der Editor-Bereich durch FROM= ... TO= ... eingerichtet und mit „ESC" wieder verlassen. Das Band wird vor den ersten Block eingerichtet. Mit „F1" wird das Programm gestartet.

Werden die beiden CR-Zeichen des letzten Blockes erkannt, verzweigt das Programm in den Monitor zurück, und es erscheint „<". Falls der letzte Block zerstört ist, muß man das Programm mit Reset zum Abschluß zwingen.

Nun kann man mit Taste „T" nachsehen, was im Editor-Bereich angekommen ist. Eventuell muß man den letzten „guten" Befehl suchen und dann den Inhalt aus DF-DE in E1-E2 einsetzen.

Reiner Kluth

Label-Sprünge beim OSI-Basic

In Heft 22/1980 der FUNKSCHAU bemängelte Hans-Georg Joepgen an vielen Microsoft-Basic-Interpretern, daß Sprünge bzw. Unterprogrammaufrufe nur mit Zeilennummern als Zieladressen, nicht aber mit Namen (Labels, wie sie bei Assemblern und sogar programmierbaren Taschenrechnern üblich sind) zulässig sind. Im folgenden wird eine Methode angegeben, wie dies beim OSI-Challenger 1P umgangen werden kann (*Bild*).

Der Ort, an dem diese Beschränkung eingebaut ist, ist die „GOTO"-Routine im ersten Basic-ROM (A6B9). Sie beginnt mit der Subroutine A77F, die in der laufenden Basic-Zeile eine 16-bit-Zahl liest. Sie prüft dann, ob die Nummer der neuen Zeile höher ist als die der laufenden Zeile; wenn ja, dann fängt sie hier an, nach der neuen Zeile zu suchen, sonst am Anfang des Basic-Programmraumes. Diese Prüfung ist aber nicht vollständig, weil sie das Low-Byte der Zeilennummer nicht einbezieht; ihr einziger Sinn ist die Einsparung von Rechenzeit.

Der neue Weg geht über die Routine AAAD, die in der Zeile einen beliebigen Ausdruck interpretiert. Die Routine B408 bildet daraus ein 16-bit-Integer. Wie die alte GOTO-Routine sucht auch die neue Routine nach der Zeilennummer mit Hilfe der Routine A432 (*Tabelle*).

Hier ist eine weitere Änderung nötig. Will man nämlich eine komplizierte Berechnung anstellen, um das Sprungziel zu definieren, dann kann das Ergebnis knapp die Zeile verfehlen, die man meint. GOTO 3999.99 ergibt einen Sprung zur Zeile 3999. Die Routine A432 kommt mit gesetztem Carry zurück, wenn die Suche erfolgreich war. Microsoft-Basic zweigt hier ab zur Fehlermeldung „US Error", wenn das Carry nicht gesetzt ist. Wer die Prüfung etwas aufwendiger gestalten will, kann stattdessen nachsehen, ob A432 am Ende des Programms angelangt ist – was ja heißt, daß die gemeinte Subroutine nicht existiert. Der Trick besteht darin, daß A432 in den Speicherstellen AA und AB die Adresse der gesuchten oder ersatzweise der nächsthöheren Zeile zurückreicht. An dieser Adresse liegt ein Zeiger auf die nächstfolgende Zeile. Wenn man am Ende des Programms ist, gibt es keine nächste Zeile – der Zeiger steht dann auf Null. In diesem Fall kann man zur Fehlermeldung springen. Dieses etwas aufwendigere Prüfverfahren braucht den Platz, den die oben erwähnte Prüfung (ob die neue Zeile eine höhere Nummer hat als die alte) einnahm. Der Zeitaufwand für ein GOTO verdoppelt sich dadurch im Schnitt. Ein weiterer Trick ist notwendig, aber gleichfalls einfach: Durch das mögliche ungenaue „Zielen" wird die Fehlersuche schwieriger. Man kann nun die Routine B95E benutzen, die die Nummer der Zeile ausdruckt, die in A- und X-Register steckt. (Man kann diese Routine sehr einfach für eine Trace-Funktion verwenden, wenn man den Control-C-Vektor auf ein kurzes Programm richtet, das erst B95E ruft und dann nach FF9B springt – dort sitzt die Routine, die nach jedem Basic-Befehl die Control-C-Taste abfragt, um einen Programmabbruch zu ermöglichen.) B95E druckt also die Zeilennummer der gesuchten Zeile – wer will, kann sie in Klammern setzen –, danach kommt dann die eigentliche Fehlermeldung. Dieses kurze Extra hat keinen Platz mehr in der GOTO-Routine; glücklicherweise gibt es noch bei A38C einige freie Bytes.

```
A6B9 20 AD AA JSR AAAD
A6BC 20 08 B4 JSR B408
A6BF 20 32 A4 JSR A432
A6C2 A5 AB    LDA AB
A6C4 85 C4    STA C4
A6C6 A5 AA    LDA AA
A6C8 85 C3    STA C3
A6CA D0 02    BNE A6CE
A6CC C6 C4    DEC C4
A6CE C6 C3    DEC C3
A6D0 A0 02    LDY #02
A6D2 B1 C3    LDA (C3),Y
A6D4 D0 0F    BNE A6E5
A6D6 88       DEY
A6D7 D0 F9    BNE A6D2
A6D9 A9 5B    LDA #5B
A6DB 20 E5 A8 JSR A8E5
A6DE A5 12    LDA 12
A6E0 A6 11    LDX 11
A6E2 4C 8C A3 JMP A38C
A6E5 60       RTS

A38C 20 5E B9 JSR B95E
A38F A9 5D    LDA #5D
A391 20 E5 A8 JSR A8E5
A394 4C F7 A6 JMP A6F7
```

Ersetzt man das Basic-1-ROM im Challenger 1P durch ein EPROM, so kann man leicht diese Routine einbauen, die es gestattet, Sprünge mit Labels auszuführen

Die GOTO-Routine wird auch von anderen Befehlen als Subroutine benutzt: GOSUB, RUN und THEN. RUN fällt für unsere Zwecke aus, weil es erst alle Variablen löscht, die als Sprungziel dienen könnten – RUN PEEK (8000) funktioniert zwar, ist aber normalerweise sinnlos. THEN nimmt den Ausdruck nur dann als Sprungziel, wenn er mit einer Zahl beginnt. Die besten Möglichkeiten bieten GOTO und GOSUB. Jede Variable kann allein oder mit beliebigen mathematischen Ausdrücken das Sprungziel definieren. Etwa den Befehl

INPUT A: ON A GOTO 10020, 10040, 10060, 10080

kann man jetzt ersetzen durch

INPUT A:GOTO A*20 + 10000

Das ist nicht nur übersichtlicher und kürzer, sondern kann Hunderte von Sprungzielen ansteuern. Weil Microsoft-Basic beliebig lange Variablen-Namen zuläßt, wobei nur die ersten beiden Buchstaben oder Ziffern interessieren, kann man die Sprungziele jetzt mit sinnvollen Eigennamen versehen. Dazu gehört dann ein Deklarationsteil am Anfang des Programms – den man erst schreibt, wenn die Sprungziele alle bekannt sind –, in dem die Sprungziele an Variablen zugewiesen werden. Das liest sich dann so:

```
10 EINGABE = 300:DRUCKEN = 30000
300 INPUT A:GOSUB DRUCKEN: GOTO EINGABE
30000 PRINT A: RETURN
```

Es gibt zwei Möglichkeiten, die Änderungen in den Computer zu implementieren. Die erste Variante ist es, das Basic-1-ROM gegen ein EPROM auszutauschen. Dabei kann man gleichzeitig die lästigen Grafik-Zeichen bei den Fehlermeldungen durch richtige Buchstaben ersetzen. Das gesetzte High-Bit bei den Fehlermeldungen hat nämlich ausschließlich historische Gründe. (Bei der Disk-Version, die ebenfalls mit verändertem GOTO arbeiten kann, gingen die OSI-Leute sogar soweit, bei allen zu druckenden Zeichen das High-Bit wegzustreichen, statt die Fehlermeldungen auszubessern.

Die zweite Methode führt über die USR-Funktion. Will man nicht viel Aufwand treiben, dann kann man die hier gezeigte Routine direkt als USR-Routine verwenden; allerdings ist sie nur als GOTO zu gebrauchen. In der hier gezeigten Form lautet die Syntax X = USR(X) Ausdruck. Das X wird dabei nicht interpretiert.

Wolfgang Baer

Tabelle:
ROM-Adressen im Challenger 1P

A6B9	GOTO-Routine
A77F	Aus der lfd. Basic-Zeile 16-bit-Zahlen lesen
AAAD	Ausdruck interpretieren
B408	Integerzahl bilden (16 bit)
A432	Zeilenadresse suchen
A8E5	Zeichen im Akku drucken
B95E	Nummer der Zeile in A und X drucken
A6F7	Fehlermeldung „US Error" drucken

Michael Schlierbach

Erweiterter Zeichensatz beim AIM-Drucker

Ursprünglich ist der in den Mikrocomputern AIM-65 und PC-100 eingebaute Thermodrucker nur für die Verarbeitung von 6-bit-ASCII-Zeichen vorgesehen; der Ausdruck z. B. von Kleinbuchstaben oder gar Umlauten ist nicht möglich. Abhilfe schafft das hier vorgestellte Programm, das mit dem ROM-Texteditor zusammenarbeitet.

An und für sich ist der AIM-65 ein recht komfortables Gerät: Der Monitor enthält einen Texteditor, ein Programm zur Ansteuerung eines kleinen Thermodruckers und einer alphanumerischen Anzeige. Allerdings ist der mögliche Zeichensatz für den Editor etwas unbefriedigend, und auch die Zeichen, die sich mit dem Drucker darstellen lassen, reichen nicht immer aus. Mit dem folgenden Programm wird die Eingabe in den Editor und der Ausdruck aus dem Druckerbuffer, bzw. das Listen aus dem Editor mit dem Drucker, auf 128 Zeichen im Bereich von hex 20 bis 9F erweitert. Die Zeichengestaltung wird dabei von einem Zeichengenerator im RAM übernommen, der auch nach eigenen Vorstellungen programmiert werden kann.

Das Programm ist in *Bild 1* aufgelistet. Es besteht weitgehend aus modifizierten Monitorroutinen. *Bild 2* zeigt die Belegung des Zeichengenerators mit der sich der in *Bild 3* gezeigte Zeichensatz ergibt.

Eingabe über die Tastatur

Um die 128 möglichen Zeichen auch gut eingeben zu können, wurde die Eingabe im Editor etwas modifiziert und ist über IN=UU zu erreichen. Will man also den neuen Zeichensatz im Editor verwenden, so ist das nur nach dem EDITOR- oder READ-Befehl möglich. FIND, CHANGE und INSERT funktionieren nur mit dem regulären Zeichensatz. INSERT läßt sich aber durch den READ-Befehl ersetzen, der die einzugebende(n) Zeile(n) vor der gerade aktuellen Zeile einfügt.

Befindet man sich in der User-Input-Routine, so kann man 125 von den 128 möglichen Zeichen des neuen Zeichensatzes direkt über die Tastatur eingeben. Die drei Ausnahmen sind hex 7F, 9C und 9F. Alle anderen Zeichen lassen sich nach dem Schema in *Bild 4* eingeben.

Während der Eingabe wird der Code der ersten zwanzig Zeichen einer Zeile auf einen von der Anzeige darstellbaren Wert gebracht, so daß man eine ungefähre Kontrolle hat, ob das richtige Zeichen eingegeben wurde. Wird später eine Zeile im Editor zur Anzeige gebracht, so sind nur die normalen Zeichen zu sehen, da die Anzeige Werte, die nicht im Bereich von 20 hex bis 5F hex liegen, als Leerzeichen ausgibt. Die User-Input-Routine befindet sich auf den Speicherplätzen 0D0F bis 0D75. Der Wert für den User-Vektor ist 0D12.

Der Zeichensatz

Der Zeichensatz hält sich weitgehend an die ASCII-Vereinbarungen. Sonderzeichen im Bereich von 80 hex bis 9F hex wurden hinzugefügt.

Um die Eingabe dieser Sonderzeichen nicht allzu kompliziert zu machen, wurden sie so auf der Tastatur verteilt, daß die Buchstaben auf den Tasten als Orientierungshilfe dienen. So ist etwa das Pi-Symbol über die Taste P plus Shift rechts zugänglich, ebenso erreicht man das Quadrat-Symbol durch Drücken von Q und Shift rechts und das Herz mit H und Shift rechts.

Ausdruck aus Editor und Buffer

Der Ausdruck des neuen Zeichensatzes aus dem Editor erfolgt im LIST-Befehl mit OUT=U. Dabei kann der Abstand zwischen den einzelnen Zeilen geändert werden. Der entsprechende Wert steht in Zelle 0BDC. Der geringste Abstand, bei dem die 5 x 8-Matrix-Zeichen direkt aneinanderstoßen, wird mit 50 hex vorgegeben. Dies ist vor allem für die Erstellung von Balkendiagram-

```
[M]=0BB0 00 00 00 00 00 2C 11 A4 10 2A 20 CB F0 20 1B 0C
[ ] 0BC0 A9 C1 8D 0C A8 20 A0 FF D0 08 20 A0 FF D0 03 4C
[ ] 0BD0 79 F0 20 E5 0B 20 E5 0B AD 77 A4 C9 5A 90 F3 A9
[ ] 0BE0 E1 8D 0C A8 60 A9 00 8D 01 A8 AD 0D A8 29 02 F0
[ ] 0BF0 F9 AD 0C A8 49 01 8D 0C A8 EE 77 A4 AD 79 A4 0D
[ ] 0C00 00 A8 8D 00 A8 AD 78 A4 8D 01 A8 A9 A4 8D 08 A8
[ ] 0C10 A9 08 8D 09 A8 20 1B 0C 4C BA F0 A2 00 20 4D 0C
[ ] 0C20 BC 60 A4 B1 00 2C 7C A4 F0 16 AD 7A A4 F0 08 0D
[ ] 0C30 78 A4 8D 78 A4 D0 09 AD 7B A4 0D 79 A4 8D 79 A4
[ ] 0C40 0E 7A A4 2E 7B A4 CA CA 10 D6 4C 18 F1 BD 74 A4
[ ] 0C50 10 1E BD 75 A4 F0 05 DE 75 A4 10 33 BD 76 A4 F0
[ ] 0C60 0A DE 76 A4 A9 04 9D 75 A4 10 24 FE 74 A4 10 1C
[ ] 0C70 BD 75 A4 C9 04 F0 05 FE 75 A4 10 13 BD 76 A4 D0
[ ] 0C80 08 9D 75 A4 FE 76 A4 10 06 DE 74 A4 1E 7C A4 A9
[ ] 0C90 00 9D 78 A4 9D 79 A4 9D 7B A4 A9 01 9D 7A A4 BD
[ ] 0CA0 75 A4 0A A8 B9 76 0D 85 00 B9 77 0D 85 01 A9 12
[ ] 0CB0 0D 76 A4 AA 60 48 20 9E EB C9 0D F0 07 AE 16 A4
[ ] 0CC0 E0 14 D0 16 48 A9 00 AE 16 A4 8D 16 A4 20 38 F0
[ ] 0CD0 20 B5 0B A2 00 68 C9 0D F0 0E 9D 60 A4 EE 16 A4
[ ] 0CE0 E8 29 80 D0 03 20 38 F0 20 AC EB 68 60 A9 20 E0
[ ] 0CF0 14 F0 06 9D 60 A4 E8 10 F6 60 A9 0D 4C 00 F0 90
[ ] 0D00 F9 38 6E 11 A4 68 08 20 B5 0C 28 2E 11 A4 60 20
[ ] 0D10 DC E7 20 37 0D C9 7F D0 06 88 10 F3 C8 F0 F3 C9
[ ] 0D20 0D F0 13 48 C9 5F 90 0A 85 00 24 00 10 02 09 40
[ ] 0D30 29 5F 20 05 EF 68 60 8A 48 AE 15 A4 E0 20 B0 05
[ ] 0D40 A9 DE 20 7B EF 68 AA 20 9E EB 20 40 EC 20 54 0D
[ ] 0D50 20 AC EB 60 48 8A D0 0C 68 48 29 40 F0 04 68 09
[ ] 0D60 20 48 68 60 29 50 F0 FA 29 10 F0 05 68 18 69 30
[ ] 0D70 60 68 18 69 40 60 60 0D E0 0D 60 0E E0 0E 60 0F

[M]=0108 12 0D FF 0C
```

Bild 1. Hex-Dump des Steuerprogramms. Es erweitert den Zeichensatz des AIM-65/PC-100-Thermodruckers um vom Benutzer selbst definierbare Zeichen, die als „User Character Generator" im Adressenbereich 0D80...0FFF abgelegt werden müssen

```
[M]=oD8o oo oo oo 14 24 63 6o oo oo oo 14 o8 4o o8 4o 6o
[ ] oD9o 3E 44 62 41 18 27 3C o1 36 46 oo 4o o8 14 41 o2
[ ] oDAo 3E 7E 7F 3E 7F 7F 7F 3E 7F oo 2o 7F 7F 7F 7F 3E
[ ] oDBo 7F 3E 7F 46 o1 3F o7 7F 63 o7 61 7F o3 oo o4 oo
[ ] oDCo FF 38 7F 38 38 38 7E 98 7F oo oo 7F oo 78 7C 38
[ ] oDDo FC 38 oo 48 o4 3C oC 3C 44 o4 44 38 oo 38 3C 7F
[ ] oDEo 3E 79 6o 55 o8 63 o8 o8 oC 14 1F o1 F8 22 o1 3D
[ ] oDFo o8 oD 2o FE 8o 3D o8 41 45 oo 5C oo o4 oo 1o 8o
[ ] oEoo oo oo o7 7F 2A 13 4E o4 1C 41 o8 o8 3o o8 oo 1o
[ ] oE1o 41 42 51 41 14 45 4A 71 49 49 oo 34 14 14 41 o1
[ ] oE2o 41 o9 49 41 41 49 o9 41 o8 41 4o o8 4o o2 o6 41
[ ] oE3o o9 41 o9 49 o1 4o 18 2o 14 o8 51 41 o4 oo o2 oo
[ ] oE4o FF 44 28 44 44 54 o9 A4 o8 oo oo 1o 3F o4 o8 44
[ ] oE5o 28 44 78 54 3F 4o 3o 4o 28 88 64 45 7F 45 41 oo
[ ] oE6o 3E 14 5o 55 o8 55 o8 1C 12 14 o1 o1 8o 14 7F 44
[ ] oE7o 78 oD 4o o1 8o 4o 14 5D 29 2o 32 o8 o4 41 2o 8o
[ ] oE8o oo 4F oo 14 7F o8 59 o2 22 22 3E 3E oo o8 oo o8
[ ] oE9o 41 7F 51 49 12 45 49 o9 49 49 44 oo 22 14 22 51
[ ] oEAo 5D o9 49 41 41 49 o9 41 o8 7F 41 14 4o oC o8 41
[ ] oEBo o9 51 19 49 7F 4o 6o 18 o8 78 49 41 o8 41 7F 7F
[ ] oECo FF 44 44 44 44 54 o9 A4 o4 7A 8o 28 4o 78 o4 44
[ ] oEDo 44 44 o4 54 44 4o 4o 3o 1o 7o 54 44 oo 44 4o 7F
[ ] oEEo 3E 12 48 55 2A 49 2A 2A 24 3E o1 o1 8o o8 o1 44
[ ] oEFo o8 oA 3F 49 8o 4o 22 22 11 77 2A 36 1C 36 7F 8o
[ ] oFoo oo oo o7 7F 2A 64 26 o1 41 1C o8 o8 oo o8 oo o4
[ ] oF1o 41 4o 49 55 7F 45 49 o5 49 29 oo oo 41 14 14 o9
[ ] oF2o 55 o9 49 41 22 49 o9 49 o8 41 3F 22 4o o2 3o 41
[ ] oF3o o9 21 29 49 o1 4o 18 2o 14 o8 45 oo 1o 41 o2 oo
[ ] oF4o FF 28 44 44 28 54 o2 98 o4 oo 7A 44 4o o4 o4 44
[ ] oF5o 44 28 o4 24 4o 4o 3o 4o 28 o8 4C 29 7F 45 41 oo
[ ] oF6o 3E 14 5o 55 o8 41 1C o8 12 14 o1 o1 8o 14 7F 44
[ ] oF7o 78 oo o1 49 8o 4o 14 14 29 o2 26 41 1o o8 2o 8o
[ ] oF8o oo oo oo 14 12 63 5o oo oo oo 14 o8 oo o8 oo o3
[ ] oF9o 3E 4o 46 22 1o 39 31 o3 36 1E oo oo 41 14 o8 o6
[ ] oFAo 1E 7E 36 22 1C 41 o1 7A 7F oo o1 41 4o 7F 7F 3E
[ ] oFBo o6 5E 46 31 o1 3F o7 7F 63 o7 43 oo 6o 7F o4 oo
[ ] oFCo FF 7C 38 oo 7F o8 oo 7C 78 oo oo 44 oo 78 78 38
[ ] oFDo 38 FC oo oo oo 3C oC 3C 44 o4 44 7C oo 38 3C 7F
[ ] oFEo 3E 79 6o 55 o8 41 o8 o8 oC 14 o1 1F 8o 22 o1 3D
[ ] oFFo o4 oo o1 36 F8 3D o8 o8 45 oo 1D oo 1o oo 1o 8o
```

Bild 2. RAM-Zeichengenerator mit Kleinschreibung, deutschen Umlauten, mathematischen Sonderzeichen und Grafiksymbolen

men mit dem Füllzeichen (60 hex) vorteilhaft. Der größte Abstand mit 18 Punktreihen zwischen den Zeilen wird mit FE hex erreicht. Standardwert in 0BDC ist 5A hex. Auch die Intensität des Druckes kann verändert werden. Um dies zu erreichen, kann man den Wert in 0C11, der normalerweise 06 ist, von 00 (sehr hell) bis maximal 0A (sehr dunkel) einstellen.

Die User-Output-Routine residiert von 0CB5 bis 0D0E. Der Wert für den User-Vektor ist 0CFF. Um nur den augenblicklichen Bufferinhalt ab A460 auszudrucken, muß das Unterprogramm ab 0BB5 bei eingeschaltetem Drucker aufgerufen werden.

Der Zeichengenerator

Um neue Zeichen programmieren zu können und den alten Zeichensatz zu erweitern, mußte der Charaktergenerator auf RAM-Zellen gelegt werden. Dazu wurden einige Routinen aus dem Monitor-ROM modifiziert. Diese waren INCP, IPST und PRTDOT (vgl. 65XX Micro Mag). Der Charaktergenerator befindet sich in den RAM-Zellen 0D80 bis 0FFF. Die Anfangsadressen für die Spalten 1–5 jedes Zeichens sind 0D80, 0E00, 0E80, 0F00 und 0F80. Das jeweilige Generatorbyte für ein Zeichen errechnet sich aus dem Code des Zeichens plus dem entsprechenden Wert aus der Vektorentabelle ab 0D76. So ist zum Beispiel das Generator-Byte für die 4. Spalte des Zeichens A die Zelle 0F21.

Abschließend sollte noch gesagt werden, daß diese Erweiterung sich nicht nur in den Editor-Befehlen E, R und L verwenden lassen sollte. Vielleicht findet der Leser noch eine gute Möglichkeit, die Erweiterung auch für die Befehle F und C zugänglich zu machen.

Auch an die Kopplung mit einem Programm, welches drei Druckerstreifen zum Nebeneinanderkleben produziert, wäre zu denken. Dann könnte auch die Zeilenbreite von 60 Zeichen ausgenutzt werden.

Um Eingabefehler bei den langen Hex-Listings in Bild 1 und Bild 2 zu vermeiden, zeigt *Bild 5* Prüfsummen für unterschiedliche Adressenbereiche. Es wird dringend geraten, diese in Heft 2/1981 näher beschriebene Technik nebst dem dort veröffentlichten Prüfprogramm anzuwenden, da der Drucker leicht durch eine falsche Ansteuerung – wie sie durch Tippfehler im Listing möglich wäre – beschädigt werden kann.

```
[M]FROM=oBB5 TO=oD8o   CAC4
[M]FROM=oD8o TO=1ooo   827o
[M]FROM=oBB5 TO=1ooo   4D34
```

Bild 5. Es empfiehlt sich, die korrekte Eingabe der Hex-Listings mit der in Heft 2/1980 beschriebenen Prüfsummen-Methode zu testen

	MSD 2	3	4	5	6	7	8	9
LO		0	@	P	▮	p	■	⌘
S1	!	1	A	Q	a	q	Ä	²
D2	"	2	B	R	b	r	Δ	√
3	#	3	C	S	c	s	≡	ß
4	$	4	D	T	d	t	÷	⌟
5	%	5	E	U	e	u	Σ	ü
6	&	6	F	V	f	v	→	◇
7	'	7	G	W	g	w	←	▷
8	(	8	H	X	h	x	♡	x̄
9	)	9	I	Y	i	y	≠	≠
A	*	:	J	Z	j	z	⌜	φ
B	+	;	K	[	k	ä	¬	{
C	,	<	L	\	l	‖	⌞	↴
D	−	=	M	]	m	ö	×	}
E	.	>	N	↑	n	ü	⌝	↓
F	/	?	O	←	o	‖	ö	–

Bild 3. Hexadezimale Codes der neuen Zeichen. Die Belegung entspricht weitgehend der ASCII-Norm

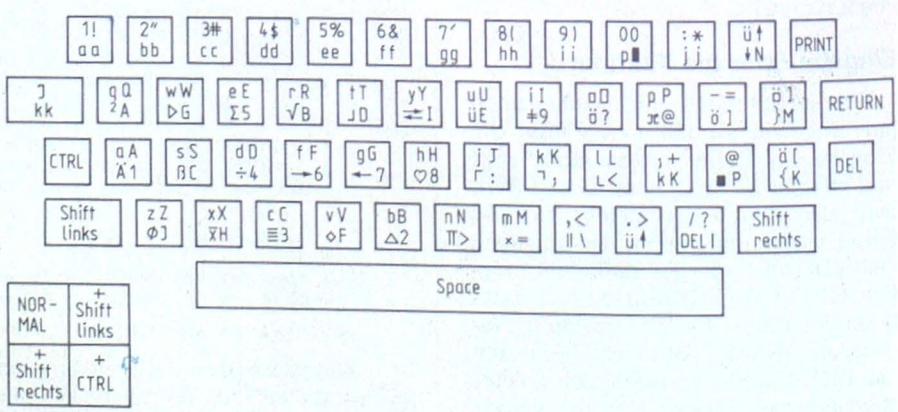


Bild 4. So sind die neuen Zeichen auf dem Tastenfeld angeordnet. Man beachte, daß die linke und rechte Shift-Taste jetzt eine unterschiedliche Funktion haben; anders wäre ein Zugriff auf (fast) alle 128 Zeichen kaum möglich

Vertikal drucken mit dem AIM-65

Zum Disassemblieren von Objektcodes eignet sich der Thermodrucker des AIM-65 (PC-100) in idealer Weise. Soll jedoch ein Text geschrieben werden, ergibt sich aufgrund der auf 20 Zeichen begrenzten Zeilenbreite eine schlecht zu lesende Darstellung. Durch das vorliegende Programm wird es möglich, pro Programmdurchlauf zehn Zeilen mit bis zu 120 Zeichen seitwärts auszudrucken (*Bild 1*).

Der zu druckende Text muß im Editor stehen und darf zwischen 1 und 99 Zeilen lang sein. Beim Initialisieren des Editors ist darauf zu achten, daß sich die Adreßbereiche des Editorbuffers und des Druckprogrammes nicht überschneiden.

```
<I>
FIRST LINE=01
DOUBLE? N
                                    DIESES PROGRAMM ERLAUBT ES, MIT DEM AIM-65 ODER PC-100 EINE
                                    GROESSERE DRUCKBREITE ZU ERZIELEN, INDEM DIE IM TEXTEDITOR
                                    STEHENDEN ZEILEN QUER AUSGEGEBEN WERDEN.
                                    DAS PROGRAMM DRUCKT DABEI BIS ZU ZEHN TEXTZEILEN MIT BIS ZU
                                    120 ZEICHEN BLOCKWEISE. DAMIT WIRD EIN HAUPTNACH-
                                    TEIL DES AIM-DRUCKERS, NAEMLICH DIE GERINGE DRUCKBREITE,
                                    BEHOBEN.
```

Bild 1. Durch geschickte Ansteuerung der Drucker-Heizelemente lassen sich Textzeilen aus dem AIM-65/PC-100-Editor quer ausdrucken, so daß sich ein breiteres Format ergibt

Das Programm (*Bild 2*) wird bei Schalterstellung „RUN" mit der Taste „F1" gestartet. In der Anzeige erscheint nun die Frage „FIRST LINE=". Darauf muß eine zweistellige Dezimalzahl eingetippt werden, welche auf die erste zu druckende Editorzeile zeigt. Als zweite Frage erscheint „DOUBLE?" in der Anzeige. Wird mit „N" geantwortet, entspricht jede Editorzeile einer Druckzeile. Ein „Y" bedeutet, daß jeweils zwei nacheinanderstehende Editorzeilen zu einer Druckzeile zusammengefaßt werden. Da das Monitorprogramm die Editorzeilenlänge nur auf 60 Zeichen beschränkt, können somit in diesem Modus bis zu 120 Zeichen pro Druckzeile dargestellt werden.

Mit dieser Eingabe beginnt auch der Druckvorgang. Sofern er nicht mit „ESC" vorzeitig abgebrochen wird, dauert die Prozedur, bis das letzte Zeichen der längsten Zeile gedruckt ist. Zum Programm: Beim gewohnten waagrechten Drucken erhält der Computer vom Druckwerk Steuersignale, sobald die Heizelemente die Position eines jeden Druckpunktes (Dots) erreicht haben. Da diese Positionen für das Vertikaldrucken ungeeignet sind, werden, ausgehend von diesen Signalen, neue Positionen extrapoliert. Somit ist man in weiten Grenzen unabhängig von Schwankungen der Motorgeschwindigkeit.

Die Heizelemente des Druckkopfes sind jeweils nur 1,7 ms eingeschaltet. Eine zu lange Einschaltzeit könnte den Druckkopf beschädigen. Im „STEP"-Betrieb ist die Einschaltzeit zu lang; dieser ungünstige Sachverhalt wird im vorliegenden Programm erkannt und es erfolgt sofort der Rücksprung ins Monitorprogramm. Ein Tippfehler beim Eingeben des im Bild stehenden Objektcodes hätte ein unvorhergesehenes Verhalten in der Programmführung zur Folge. Die korrekte Funktion sollte deshalb mit reduzierter Druckspannung kontrolliert werden. Dies wird erreicht durch Drehen des Trimmpotentiometers VR 2 im Uhrzeigersinn. Arbeitet das Programm einwandfrei, kann das Potentiometer wieder in die Ausgangsstellung zurückgedreht werden.

Matthias Meier

```
0D56 A2 0A A9 00 8D 22 A4 4D 22 A4 F0 01 60 8D 0E A0
0D66 8D 0B A0 8D 0B A8 20 13 EA BD F5 0F 20 7A E9 CA
0D76 10 F7 20 FD E3 8D 19 A4 A2 07 20 13 EA BD ED 0F
0D86 20 7A E9 CA 10 F7 20 73 E9 A2 00 C9 59 F0 08 A2
0D96 80 20 13 EA 20 13 EA 8E 7B A4 A9 FF 8D 7C A4 20
0DA6 BC F8 4C AE 0D 20 22 0E 20 90 E7 D0 F8 20 34 F9
0DB6 A2 09 AD 7B A4 9D 60 A4 CA 10 F7 A2 00 EE 7C A4
0DC6 20 D0 F8 AC 7C A4 A9 40 3D 60 A4 D0 21 BD 60 A4
0DD6 4D 7B A4 10 01 C8 A9 80 3D 60 A4 9D 60 A4 B1 DF
0DE6 F0 48 C9 0D F0 25 29 3F 1D 60 A4 9D 60 A4 20 22
0DF6 0E 2C 7B A4 30 03 20 22 0E E8 E0 0A 90 C5 A9 E0
0E06 CA 30 1F DD 60 A4 F0 F8 4C 3A 0E BD 60 A4 30 05
0E16 C8 A9 80 D0 C6 A9 E0 9D 60 A4 D0 D2 A0 00 20 13
0E26 F7 60 A2 00 20 38 F0 4C 4A F0 A9 E0 9D 60 A4 E8
0E36 E0 0A 90 F8 A9 00 8D 7A A4 8D 77 A4 20 9A 0F EE
0E46 77 A4 A9 C0 8D 0C A8 AD 00 A8 20 A0 FF D0 08 20
0E56 A0 FF D0 03 4C 79 F0 20 45 0F A9 FF 8D 08 A0 8D
0E66 09 A0 20 58 0F 20 45 0F 20 26 0F 20 19 0F 20 14
0E76 0F 20 58 0F 20 4D 0F 20 26 0F 20 19 0F A0 02 20
0E86 88 0F 20 4D 0F 20 19 0F 20 58 0F EE 7A A4 20 05
0E96 0F 20 14 0F 20 58 0F AD 7A A4 C9 05 90 13 A0 13
0EA6 20 45 0F 88 10 FA A9 E1 8D 0C A8 20 07 E9 4C C1
0EB6 0D 20 45 0F A9 FF 8D 08 A0 8D 09 A0 20 58 0F 20
0EC6 45 0F 20 26 0F 20 45 0F 20 58 0F 20 45 0F 20 19
0ED6 0F 20 26 0F 20 14 0F A0 01 20 88 0F 20 45 0F 20
0EE6 19 0F 20 58 0F 20 14 0F 20 58 0F A9 00 8D 77 A4
0EF6 EE 7A A4 20 05 0F 20 45 0F 20 58 0F 4C 5D 0E 18
0F06 AD 7D A4 69 40 8D 7D A4 90 03 EE 7E A4 60 2C 0D
0F16 A0 50 FB AE 76 A4 AD 75 A4 8D 04 A0 8E 05 A0 60
0F26 A9 FF 38 ED 08 A0 AA A9 FF ED 09 A0 8D 76 A4 8A
0F36 C9 08 B0 03 CE 76 A4 4E 76 A4 6A 8D 75 A4 60 AD
0F46 0C A8 49 01 8D 0C A8 AD 01 A8 A9 02 2C 0D A8 F0
0F56 FB 60 AD 79 A4 0D 00 A8 8D 00 A8 AD 78 A4 8D 01
0F66 A8 A9 A4 8D 08 A8 A9 06 8D 09 A8 20 A4 0F EE 77
0F76 A4 20 1B EC A9 00 8D 01 A8 AD 00 A8 29 FC 8D 00
0F86 A8 60 8C 74 A4 20 58 0F 20 14 0F 20 58 0F CE 74
0F96 A4 10 F5 60 A9 E1 8D 7D A4 A9 F2 8D 7E A4 AD 77
0FA6 A4 F0 24 C9 08 F0 26 B0 0F A2 09 5E 6A A4 2E 78
0FB6 A4 2E 79 A4 CA 10 F4 60 A2 09 1E 6A A4 2E 78 A4
0FC6 2E 79 A4 CA 10 F4 60 20 D9 0F 4C AF 0F 20 D9 0F
0FD6 4C BE 0F A2 09 BD 60 A4 29 3F A8 A9 7D 20 58 EB
0FE6 9D 6A A4 CA 10 EF 60 20 3F 45 4C 42 55 4F 44 3D
0FF6 45 4E 49 4C 20 54 53 52 49 46
```

Bild 2. Das Programm belegt weniger als 1 KByte. Es empfiehlt sich, die Funktion zuerst bei verringerter Druckerspannung zu prüfen. Die hexadezimale Prüfsumme aller Programmbytes (siehe Prüfsummen-Programm in Heft 2/1981, Seite 84) ist 1C36

Basic-Variablenliste

Das in *Bild 1* abgedruckte Basic-Programm kann für ein beliebiges anderes Basic-Programm (oder sich selbst) eine Liste ausgeben, in welchen Zeilen bestimmte Variablennamen verwendet werden. Es wurde für den TRS-80 Level II entwickelt und läßt sich daher nicht ohne weiteres z. B. auf den PET-2001 übertragen.

Die Ausgabe erfolgt in alphabetischer Reihenfolge der Variablen; erscheint eine Variable innerhalb einer Zeile mehrfach, so wird die entsprechende Zeilennummer wiederholt ausgegeben.

Einen Beispielausdruck zeigt *Bild 2*; hier hat sich das Programm selbst analysiert. Es unterscheidet nicht zwischen Feldvariablen gleichen Namens und gleicher Art, aber unterschiedlicher Anzahl der Dimensionen, was in Level-II-Basic ja erlaubt ist. Da dieser Fall selten vorkommt, wurde wegen des damit verbundenen höheren Programmaufwandes auf die Unterscheidung verzichtet.

```
 LISTE DER VARIABLEN:
 --------------------
A       50040 50040 50040 50040 50040 50040 50050
B       50030 50030 50040 50040 50050 50080
I       50050 50050 50080 50080 50080 50080 50180 50200 50200
        50200 50200 50200 50210 55000 55000 55000
J       50130 50130 50130 50130 50140 50150 50160 50160 50170
        50172 50172 50172 50178 50178 50178 50178
K       50150 50150 50150 50150 50150 50150 50170 50170 50172
        50175 50190 50190 50200 50210
L       50030 50130 50130 50150 50150 50150 50200
L!      50040 50040 50178
L$      50050 50080 50120 50120 50130 50140 50160 55000 55000
L$()    50010 50020 50130 50140 50150 50150 50160 50178 50200
        50200
P       50050 50050 50050 50060 50060 50070 50070 50070 50080
        50090 50100 50100 50100 50100 50110 50110 50110
        50120 55000 55000
Q       50030 50030 50040 50040
V       50070 50070 50090
Y       50030 50040 50040 50210
Y$      50210 50210 50210
Z$()    50010 50020 50150 50150 50160 50170 50172 50172 50172
        50178 50178 50178 50200 50200

SPEICHERPLATZBEDARF (STAT.) 1264 BYTES
```

Bild 2. So sieht die Variablenliste aus. Sie ist alphabetisch nach Variablennamen geordnet und gibt an, welche Variablen in welchen Programmzeilen vorkommen

Das Laden des Programms geschieht am besten folgendermaßen: Zunächst stellt man mit PRINT PEEK (16548); PEEK (16549) den Programmanfang fest. In Level II ist das gewöhnlich 233/66 (d. h. hex 42E9), in Level III dagegen 192/86 (hex 56C0). Mit PRINT PEEK (16633); PEEK (16634) kann man dann das Programmende bzw. den Beginn des Variablenbereiches feststellen, z. B. 211/71. Zwischen dem Programmende und dem Beginn der Variablenliste steht noch ein Zeilenzeiger mit 00. Von der gerade ermittelten Adresse muß man deshalb 2 abziehen, so daß sich z. B. 209/71 ergibt. Dabei ist ggf. ein hexadezimaler Übertrag zu berücksichtigen. Die sich so ergebende Adresse muß man nun als Programmanfang definieren: POKE 16548,209: POKE 16549,71.

Jetzt kann man das Variablen-Auflistprogramm mit CLOAD einlesen. Das ursprüngliche Programm wird wieder „aufgedeckt" mit: POKE 16548,233: POKE 16549,66 (unter Level III: 192/86). Der Start des Auflistens erfolgt schließlich mit RUN 50 000.

Ing. (grad.) Heinz-Peter Curdts

```
50000 REM VARLIST (C) HEINZ-PETER CURDTS (19.04.80)
50010 CLEAR 3000:DEFINT A-Z:DIM L$(150),Z$(150)
50020 L$(1)=" LISTE":Z$(1)=" DER VARIABLEN:"+CHR$(13)+" "+
      STRING$(20,"-")
50030 L=1:D=256:B=PEEK(16548)+D*PEEK(16549):Y=B
50040 A=B:B=PEEK(A)+D*PEEK(A+1):L!=PEEK(A+2)+D*PEEK(A+3):
      IF L!)49999 THEN Y=A-Y:GOTO 50190
50050 FOR I=A+4 TO B-2:P=PEEK(I):L$="":
      IF P)64 AND P(91 THEN 50100
50060 IF P=147 OR P=135 THEN 50040
50070 IF P=34 THEN V=34 ELSE IF P)151 AND P(155 THEN V=58 ELSE
      50180
50080 I=I+1:IF I)B-2 THEN 50040 ELSE P=PEEK(I):L$=""
50090 IF P()V THEN 50080 ELSE 50180
50100 GOSUB 55000:IF(P)47 AND P(58)OR(P)64 AND P(91)THEN
      GOSUB 55000
50110 IF P=33 OR(P)34 AND P(38)THEN GOSUB 55000
50120 IF P=40 THEN L$=L$+"()"
50130 FOR J=1 TO L:IF L$(=L$(J) THEN 50140 ELSE NEXT J:L=J:
      GOTO 50160
50140 IF L$=L$(J) THEN 50170
50150 FOR K=L TO J STEP-1:L$(K+1)=L$(K):Z$(K+1)=Z$(K):NEXT K:
      L=L+1
50160 L$(J)=L$:Z$(J)=""
50170 FOR K=1 TO 10:IF LEN(Z$(J))(52*K THEN 50178
50172 IF LEN(Z$(J))(=57*K THEN Z$(J)=Z$(J)+CHR$(13)+"      ":
       GOTO 50178
50175 NEXT K
50178 Z$(J)=Z$(J)+STR$(L!):PRINT L$(J);TAB(6)Z$(J)
50180 NEXT I:GOTO 50040
50190 PRINT"AUSGABE DER REFERENZLISTE:":
      PRINT"WIEDEREINSTIEG MIT 'GOTO 50200'":PRINT:
      INPUT"1=MONITOR, 2=DRUCKER, 0=END";K:IF K=0 THEN END
50200 FOR I=1 TO L:IF K=2 THEN LPRINT L$(I);TAB(6)Z$(I)
      ELSE PRINT L$(I);TAB(6)Z$(I)
50210 NEXT I:Y$=" "+CHR$(13)+"SPEICHERPLATZBEDARF (STAT.)"+
      STR$(Y)+" BYTES":IF K=2 THEN LPRINT Y$ ELSE PRINT Y$
50220 GOTO 50190
55000 L$=L$+CHR$(P):I=I+1:P=PEEK(I):RETURN
60000 REM
```

Bild 1. Programm zum Aufstellen einer Variablenliste beliebiger Basic-Programme

Berichtigung

Polarkoordinaten-Umrechnung in Basic

FUNKSCHAU 1980, Heft 26, Seite 92

Beim Abdruck des Programmlistings ist leider ein kleiner Fehler unterlaufen. In Zeile 63 muß es statt X=... richtigerweise heißen: Y=...

N. Nessler

Direkt-Assembler mit symbolischer Adressierung

Hier wird ein Direkt-Assembler für den Mikroprozessor 6502 beschrieben, der ohne Zwischenspeicherung mit geringstmöglichem Speicheraufwand sogar eine symbolische Adressierung der Sprung- und Branchbefehle zuläßt. Eine Labelliste wird automatisch generiert, eine vorausgehende Definition der Labels ist nicht erforderlich. Eine umfassende Fehlererkennung und -meldung verhindert falsche Eingaben.

Einleitung

Zur Programmierung von Mikroprozessorsystemen gibt es heute eine Vielzahl von Programmiersprachen, die sich im Komfort und der Komplexität des Übersetzerprogramms unterscheiden und mehr oder weniger gut auf bestimmte Anwendungsfälle zugeschnitten sind. Die unterste Stufe der Programmierung ist die direkte Eingabe im *Hexadezimalcode*. Es gibt keinerlei Redundanz oder Fehlererkennung. Das System ist mit einem Minimum an Hard- und Softwareaufwand betriebsbereit (z. B. KIM-1).

Die nächste Stufe stellen die Assembler dar, die zwar ebenfalls noch an einen Mikroprozessortyp gebunden sind, aber durch Eingabe im mnemonischen Code wesentlich „benützerfreundlicher" arbeiten. Man unterscheidet zwischen direkten (single pass) Assemblern, die jeden eingegebenen Befehl sofort übersetzen, und editierenden Assemblern (dual pass), bei denen das eingegebene Programm (Quellprogramm) zunächst gespeichert und dann nach allfälligen Korrekturen („Editieren") in zwei Durchläufen assembliert, d. h. in die Maschinensprache (Objektprogramm) übersetzt wird.

Die höchste Stufe der Programmiertechnik stellen die Interpreter und Compiler für höhere Sprachen dar. Letztere sind vom Mikroprozessortyp unabhängig und zumeist sehr komfortabel ausgestattet. Die Programmiersprache kann der Problemstellung angepaßt gewählt werden (z. B. Fortran, Cobol, PL/M, PLZ...). Das so maschinell übersetzte (compilierte) Objektprogramm ist aber immer wesentlich länger und komplizierter als eine direkt in Assemblersprache geschriebene Lösung. Der Vorteil liegt vor allem bei umfangreichen Programmen in der kürzeren Programmierzeit.

Alle komfortablen höheren Programmiersprachen einschließlich des editierenden Assemblers benötigen einen je nach Komfort unterschiedlichen Speicherplatz für das Übersetzerprogramm (oft bis über 20 000 Speicherworte) und einen Arbeitsspeicher für das einzugebende Programm (etwa zehn- bis 50mal umfangreicher als das endgültige Programm).

Programme mit direkter Übersetzung in Maschinensprache (auch die hexadezimale Eingabe muß in Binärzahlen übersetzt werden!) sind meist kurz (2...5 KByte) und vermeiden eine Zwischenspeicherung. Der gebotene Komfort ist daher allerdings sehr gering, eine Fehlererkennung ist bei der Eingabe überhaupt nicht vorhanden (hexadezimale Eingabe) oder auf den mnemonischen Code bechränkt. Die Adressen zu den einzelnen Befehlen müssen direkt angegeben werden, was vor allem bei Sprungbefehlen häufig zu Fehlern führt. Die Bestimmung der Sprungzieladressen erfordert eine genauere Vorbereitung des einzugebenden Programms oder eine nachträgliche Eingabe der Zieladressen.

Das im folgenden vorgestellte Assemblerprogramm ist in seinem Speicherplatzbedarf so klein wie die direkt übersetzenden, bietet aber den Komfort der speichernden Assembler. Insbesondere ist durch die Möglichkeit einer symbolischen Adressierung bei den Sprungbefehlen eine wichtige Fehlerquelle eliminiert.

Systemaufbau

Als Entwicklungssystem reicht bereits der bekannte KIM-1 mit einer leicht selbst herzustellenden Speichererweiterungsplatine aus, auf der der Assembler in PROM oder EPROM (2 K x 8 bit) resident gespeichert ist, und ein Ein-/Ausgabegerät für das Schreiben und die Kontrolle des Programms. Mit einigem Geschick läßt sich ein ausgedienter Postfernschreiber mit einer einfachen Interfaceschaltung direkt an den KIM anschließen. Die Systemkosten betragen somit nur etwa 10 % des Preises für ein kommerzielles Gerät. Steht ein ASCII-Fernschreiber oder ein Bildschirmterminal zur Verfügung, so ist ein direkter Anschluß an die dafür vorgesehene Schnittstelle des KIM möglich.

Das vorliegende Assemblerprogramm wurde für einen Postfernschreiber mit Baudotcode (CCITT Nr. 2) geschrieben. Zum Betrieb mit einem ASCII-Terminal lassen sich alle ein- und ausgegebenen Zeichen über eine Vergleichstabelle mit einem kurzen Programm umwandeln.

Arbeitsprinzip des KIM-Assemblers

Die Eingabe des Quellprogramms erfolgt im 6502-Assemblercode, dessen einfacher Aufbau sowohl die Eingabe als auch die Übersetzung sehr erleichtern. Nach dem Start des Assemblerprogramms druckt dieses eine Titelzeile und verlangt die Eingabe der Anfangsadresse in Form einer vierstelligen Hexadezimalzahl mit abschließendem Zwischenraum. Nach Initialisierung des Labelspeichers (Angabe von „E", „V" und „U", Bedeutung siehe unten) kann bereits die eigentliche Progammeingabe beginnen.

Zu Beginn jeder neuen Zeile schreibt das Programm die laufende Befehlsadresse und erwartet sodann die Eingabe des stets aus drei Buchstaben bestehenden Operationscodes. Ein Zwischenraum schließt diese Eingabe ab und startet einen Vergleich mit den in einer Tabelle gespeicherten Befehlscodes. Ein mitlaufender Suchzähler ergibt bei einer Übereinstimmung den Hinweis auf den Befehl und auf die dafür zulässige Adressierungsart. Bei Ein-Byte-Befehlen wird sofort der richtige Hexadezimalcode berechnet und an die laufende Adresse gespeichert, andernfalls werden die entsprechenden Adreß-Kennbits gesetzt. Bei falscher

Opcode-Eingabe erfolgt Fehlermeldung und Neustart dieser Zeile.

Die Adreßart Eingabe erfolgt für alle arithmetisch-logischen Befehle in hexadezimaler Schreibweise, zweistellig oder vierstellig, mit höchstwertiger Stelle zuerst. Eine symbolische Adressierung wäre hier prinzipiell möglich, würde aber den Labelspeicher wesentlich vergrößern und die reine Schreibarbeit bei der Eingabe komplizierter gestalten (Labelkennung und mehrstelliger Name), während man andererseits voraussetzen kann, daß-alle wichtigen Datenspeicheradressen in die zero page gelegt werden und nur eine zweistellige Adresse zu schreiben ist, die noch dazu Symbolcharakter haben kann (z. B. A1, A2, B0...). Die Eingabeform für die beim 6502 möglichen Adressierungsarten zeigt *Tabelle 1*.

Bereits während der Eingabe wird die Adreßart entschlüsselt und auf Vollständigkeit und Fehlerfreiheit überprüft. Die Eingabe wird durch einen Zwischenraum abgeschlossen; anschließend kontrolliert das Programm, ob die Adreßart für den vorliegenden Befehl zulässig ist. Trifft dies zu, wird der Operationscode mit der Adreßartkennung vervollständigt und der gesamte Befehl in der richtigen Reihenfolge (low-order Byte der Adresse vor high-order Byte) fortlaufend abgespeichert.

Die Weiterschaltung auf die nächste Zeile, die Berechnung und der Ausdruck der nächsten Befehlsadresse folgen automatisch und zeigen damit an, daß der Befehl richtig erkannt und assembliert wurde. Wurde fehlerhaft eingegeben (Tippfehler oder unzulässige Adressierart), so wird eine entsprechende Fehlermeldung ausgegeben und die gleiche Zeile (gleiche Adresse) automatisch nochmals neu gestartet.

Symbolische Adressierung

Die Möglichkeit, Sprung- und Branchbefehle symbolisch mit sogenannten Labels (Marken) zu adressieren und dafür weder das Quellprogramm zu speichern noch eine vorher festzulegende Labelliste zu benützen, ist der besondere Vorteil dieses Assemblerprogramms. Es ist natürlich möglich, die Sprungadresse bzw. den Offset bei Branchbefehlen direkt anzugeben, die Erfahrung mit dem KIM-Assembler zeigt aber, daß dies eher eine Ausnahme ist.

Tabelle 1:
Eingabeform der Adressierarten

Adreßart	Eingabe
zero page	NN
absolut	NNNN
immediate	#NN[1])
zero page indiziert	NN,X
absolut indiziert	NNNN,X
indirekt	(NNNN)
indirekt indiziert	(NN),Y
indiziert indirekt	(NN,X)
accumulator	AK

[1]) N steht für eine Hexadezimal-Ziffer. Im CCITT-Code ist „#" nicht vorgesehen und wird durch „=" ersetzt.

Eine symbolische Adressierung setzt voraus, daß das Sprungziel mit einem Namen (Label) bezeichnet wird, der anstelle der Sprungadresse oder des Offset *nach* dem Operationscode einzugeben ist. Die hierfür festgelegte Form ist

'NNN

wobei der Apostroph zur Unterscheidung von normaler Adressierung und symbolischer Adresse dient und im Programm den Einsprung in den Symbolteil bewirkt. Der Labelname kann aus 1...3 beliebigen FS-Zeichen bestehen (ausgenommen Klingelzeichen). Der Abschluß des Labelnamens erfolgt wieder mit Zwischenraum.

Die Festlegung des Sprungzieles, das Setzen des Labels, erfolgt, indem an die gewünschte Stelle *vor* dem Operationscode der Labelname in der oben angegebenen Form geschrieben wird: Das Programm speichert Labelname und Adresse in der Labelliste (maximal 51 Namen) und ersetzt dann jede gleichlautende symbolische Adresse durch die effektive Sprungzieladresse bzw. rechnet den entsprechenden Offset.

Dies läßt sich problemlos durchführen, solange das Label *vor* dem Aufruf dieser symbolischen Adresse gesetzt wurde, d. h., wenn es sich um einen Rückwärtssprung handelt. Wird aber im Anwenderprogramm das Label erst weiter unten definiert (Vorwärtssprung), so versagt das einfache Verfahren. Gerade dieses Problem ist die Ursache dafür, daß alle symbolisch adressierbaren Assembler zur Übersetzung 2 Durchläufe benötigen; im ersten werden alle Marken festgestellt und aufgelistet, im zweiten wird erst übersetzt.

Das vorliegende Assemblerprogramm hingegen kann auch bei späterer Definition des Labels rückwirkend einsetzen, benötigt also nur einen Durchlauf. Das rückwirkende Einsetzen von zunächst unbekannten Sprungadressen wird durch einen programmtechnischen Trick, das *Sprossenleiterprinzip*, ermöglicht. Wird bei Sprung- oder Branchbefehlen ein noch unbekanntes Label aufgerufen, so wird

1. das neue Label in der Liste zusammen mit der Adresse des Sprungbefehls gespeichert;
2. im Labelnamen eingebaut ein Kennbit gespeichert, daß es sich um ein noch nicht definiertes Label handelt,
3. und anstelle der Adresse beim Sprungbefehl FFFF, anstelle des Offset beim Branchbefehl FF gesetzt.

Diese im fertigen Programm sicher nie vorkommenden Adressen dienen als Abschlußkennung für die Sprossenleiter. Wird dieses Label noch öfter aufgerufen, bevor es effektiv definiert wird, so werden jedesmal folgende Schritte durchgeführt:

1. Die Adresse des gerade übersetzten Sprungbefehls wird als neue Adresse zum Labelnamen gesetzt und
2. die dort notierte Adresse des vorletzten Sprungbefehls wird als vorläufige Adresse zum neuen Sprungbefehl gesetzt.

Dadurch entsteht die Sprossenleiter in der Form, daß die Adresse beim Label auf die unterste Sprosse weist, dort steht jeweils die Adresse der nächsten Sprosse, usw., die oberste Sprosse enthält die Schlußadresse FFFF (*Bild 1*).

Bei Branchbefehlen steht nur ein Byte als relative Adresse zur Verfügung, hier wird innerhalb der Sprossenleiter jeweils der Offset zum vorhergehenden Befehl angegeben. Sobald das Label effektiv definiert wird, werden alle Sprungbefehle entlang der Sprossenleiter rückwirkend mit der richtigen Adresse versehen und das Kennbit gelöscht. Dieser Einsetzvorgng wird auch verwendet, wenn die Eingabe des Labels an der richtigen Stelle vergessen wurde und nachträglich mittels Zeilenkorrektur (siehe Sonderfunktionen) eingefügt wird.

Sonderfunktionen

Zur Erhöhung des Bedienungskomforts wurden im Assemblerprogramm noch einige Sonderfunktionen vorgesehen. Der Aufruf kann am Beginn jeder Zeile durch „:" (Doppelpunkt) und einen Kennbuchstaben erfolgen.

Zeilenkorrektur

„:K Zeilenadresse" nach dem Antwortausdruck „KORR" kann der zu korrigierende Befehl neu eingegeben werden

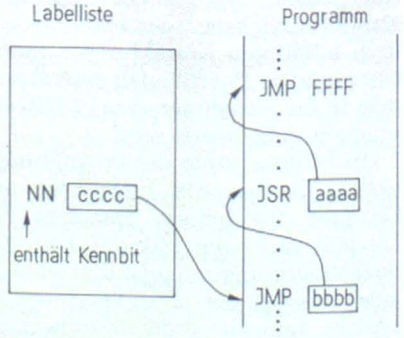


Bild 1. Prinzip der Labelliste. Das Einsetzen der Adressen wird durch das Sprossenleiter-Prinzip ermöglicht

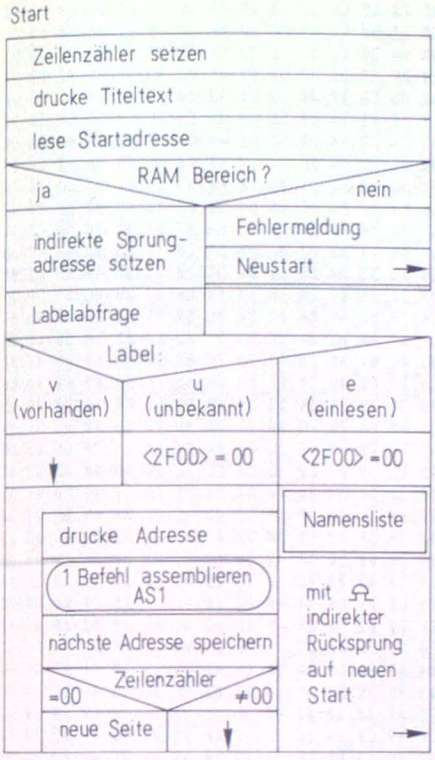


Bild 2. Struktogramm des Assembler-Hauptprogramms

oder/und nachträglich ein Label eingesetzt werden.

Programmliste
„:P" Mit einem Disassemblerprogramm (vgl. Elektronik 1979, Heft 21, S. 63–66) wird das bisher eingegebene Programm in hexadezimaler und rückassemblierter Klartextdarstellung ausgegeben.

Labelliste
„:N" Einsprung in das Namensprogramm zur Eingabe weiterer Labels in Form „aaaa Zwischenraum NNN". Mit „P" wird die bisherige Labelliste ausgedruckt, mit „Ω" (Klingelzeichen) Rücksprung ins Assemblerprogramm.

Undefinierte Label
„:U" Ausgabe aller noch nicht definierten Labels.

Fortsetzung nach Neustart
„:F" Nach verschiedenen Sonderfunktionen verlangt der Assembler die Eingabe einer neuen Startadresse. Wird hier die laufende Adresse angegeben, so erfolgt ein Ausdruck mit einem Disassembler ab dieser Adresse. Soll das gesamte Programm ausgedruckt werden, muß mit der Anfangsadresse gestartet und mit „:F" auf die laufende Adresse gesprungen werden.

Labelvorbereitung
Mit der Aufforderung LABEL wird eine Angabe zur Labelliste gefordert. Die Antwortmöglichkeiten sind „E" Einsprung in das Namensprogramm

Tabelle 2: Fehlermeldungen des Assemblers

Fehlermeldung	Bedeutung
OPCODE	falscher oder unvollständiger Operationscode
ADRESSE	Adresse unvollständig
ADRART	unzulässige Adreßart
SYNTAX	Fehler bei Sonderzeichen
LABEL	Doppeldefinition oder unzulässige Verwendung des gleichen noch nicht definitiven Labels für Sprung- und Branchbefehl
OFFSET	Sprungweite für Branchbefehl zu groß
RAM-BEREICH	Speicherbereich überschritten

analog :N, jedoch vorher Löschen aller bisherigen Labels
„U" Löschen aller Labels
„V" bereits vorhandene Labels werden berücksichtigt.

Fehlermeldungen
Bei fehlerhafter Eingabe wird in derselben Zeile eine Fehlermeldung ausgegeben, eine Speicherung des Befehls erfolgt nicht, die Zeile kann neu begonnen werden. *Tabelle 2* nennt die möglichen Fehlermeldungen.

Speicherbedarf

Das gesamte Assemblerprogramm umfaßt nur 1,5 KByte in einem PROM. Der für den Ausdruck notwendige Disassembler, der auch die Operationscode-Vergleichstabelle enthält, benötigt 1 K Byte PROM, für die Namensliste muß 1 Seite (256 Byte) RAM zur Verfügung stehen. Die Adressen 00E2 bis 00EE in der Zero Page werden für temporäre Speicher und Zähler verwendet. Das Struktogramm des KIM-Assemblers zeigt *Bild 2*.

Aus *Bild 3* geht hervor, wie die Operationscodes aus den mnemonischen Befehlen erzeugt werden. *Bild 4* zeigt schließlich das vollständige Assemblerprogramm aus Platzgründen als Hex-Dump.

Anwender, die nicht den KIM-1 verwenden, können eine Anpassung an das eigene Computersystem leicht durch Verwenden der Speicherbelegungstabelle (KIM, AIM, PET, CBM) im FUNKSCHAU-Sonderheft „Mikrocomputer-Anwendungen" vornehmen. Dabei wäre es auch leicht möglich, die Baudot-Ein-/Ausgabe durch meist schon im Monitorprogramm vorhandene ASCII-Routinen zu ersetzen, indem noch eine Baudot/ASCII-Umwandlungstabelle implementiert wird. Doch dazu später!

In *Bild 5* ist das mit dem Assembler zusammenarbeitende Disassembler-Programm aufgelistet. Der Einsprung in die Fernschreibprogramme erfolgt über die „Rettroutine" des Disassemblers, welche die Registerinhalte X, Y, P rettet und nach Rücksprung wiederherstellt; Voraussetzung ist, daß folgende Zellen belegt sind:
1780 20 00 XX
1783 4C AE 42
Die effektiven Sprünge stehen in Seite XX (nach vorhandenen Speicherplätzen frei wählbar) in der Reihenfolge:

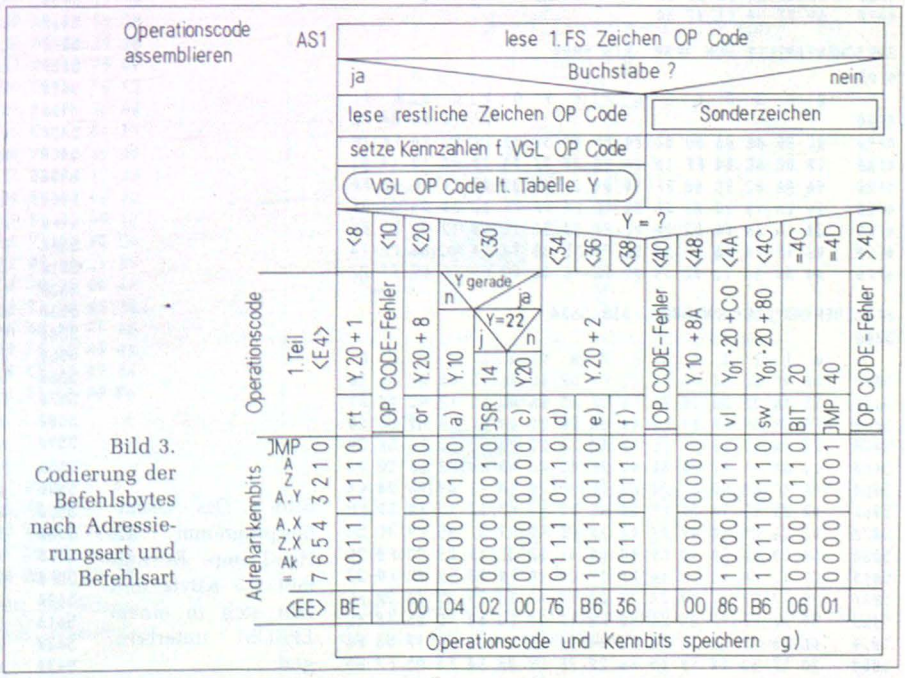


Bild 3. Codierung der Befehlsbytes nach Adressierungsart und Befehlsart

```
SPEICHERTABELLE VON 441A  BIS 4439
441A
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
4410                                  A0 10 88 30 05 D9
4420  D2 46 D0 F8 98 60 30 10 A0 04 0A 88 D0 FC A0 04
4430  0A 26 F7 26 F8 88 D0 F8 60

SPEICHERTABELLE VON 46D2  BIS 46E2
46D2
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
46D0        AC AE A6 82 94 A0 AA 8E 8C B0 06 32 1C 12
46E0  02 1A

                  4817  BIS 4826
4817
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
4810                        A0 00 B1 E2 C9 FF F0 06 20
4820  C4 42 C8 D0 F4 60

SPEICHERTABELLE VON 4A25  BIS 4A30
4A25
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
4A20                  85 E2 A9 3E 20 C4 42 20 17 48 60

SPEICHERTABELLE VON 4A6B  BIS 4BDA
4A6B
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
4A60                                  20 C7 42 18 AD
4A70  00 2F 0A 0A 6D 00 2F 85 EA E6 EA 20 90 4A EE 00
4A80  2F AD 00 2F C9 33 90 F1 A9 42 20 25 4A 6C FC 17
4A90  20 1C 4A 20 3F 49 C9 2C F0 6C C9 1E F0 5C 20 1A
4AA0  44 30 05 20 26 44 F0 EB 20 D2 4A 20 3F 49 C9 88
4AB0  F0 12 C9 08 F0 0E A2 08 0A 26 F8 26 F7 26 F6 CA
4AC0  D0 F6 F0 E7 20 C8 4A 60 A0 00 E6 EA A5 F6 91 EA
4AD0  E6 EA A0 00 A5 F7 91 EA E6 EA A5 F8 91 EA 60 20
4AE0  CA 42 A5 EB 20 C7 42 A5 EA 20 C7 42 60 20 DF 4A
4AF0  A9 03 20 C1 42 60 FF FF FF FF 68 68 A5 F7 85 EA
4B00  20 90 4A 4C 6E 4A 68 68 20 14 4B 20 CA 42 20 4A
4B10  4B 4C 6B 4A 20 45 4B 85 EC 20 C7 42 E6 EA 20 1C
4B20  4A 20 29 4B C6 EC D0 F6 60 A2 7D A0 00 B1 EA E6
4B30  EA E8 30 09 20 C7 42 20 BE 42 18 90 EE 20 C4 42
4B40  E0 82 D0 E7 60 A9 31 20 25 4A A9 2F 85 EB A9 00
4B50  85 EA 20 1C 4A AD 00 2F 60 A9 03 20 BB 42 A9 3E
4B60  20 C4 42 20 DF 4A 20 CA 42 A9 05 20 C1 42 A2 00
4B70  A9 02 20 C1 42 8A 20 F2 44 E8 E0 10 D0 F2 20 CA
4B80  42 A5 EB 20 C7 42 A5 EA 29 F0 20 C7 42 A5 EA 29
4B90  0F AA F0 08 A9 03 20 C1 42 CA D0 F8 A9 03 20 C1
4BA0  42 A1 EA 20 C7 42 20 C3 4B D0 0B 20 CA 42 A9 96
4BB0  20 C4 42 6C FC 17 A5 EA 29 0F F0 C2 20 BE 42 A2
4BC0  00 F0 DE E6 E8 D0 02 E6 E9 E6 EA D0 02 E6 EB A5
4BD0  EB C5 ED D0 04 A5 EA C5 EC 60

SPEICHERTABELLE VON 4ADF  BIS 4AF6
4ADF
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
4AD0                                                20
4AE0  CA 42 A5 EB 20 C7 42 A5 EA 20 C7 42 60 20 DF 4A
4AF0  A9 03 20 C1 42 60

SPEICHERTABELLE VON 4F8B  BIS 5000
4F8B
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
4F80                                  FF 10 04 04 1E
4F90  0C 38 08 86 08 06 0A 0A 02 38 32 24 02 14 FF 18
4FA0  18 8C AC 04 FF 10 04 0A 20 06 14 20 06 12 14 02
4FB0  0A 0A 02 9C 88 FF 10 04 24 06 32 02 24 9C 88 FF
4FC0  0A 2A 18 20 06 3A B2 FF FF FF FF 3C 30 24 24 FF
4FD0  30 1A 1A 0A 02 20 B2 FF 30 2C 1C 30 12 02 B2 FF
4FE0  06 12 14 06 14 20 B2 FF 1E 30 14 14 9C 88 FF 14
4FF0  06 38 86 32 02 14 02 0C 1C 28 B2 FF FF FF FF FF

SPEICHERTABELLE VON 5000  BIS 5634
5000
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
5000  A9 4F 85 E3 A9 32 85 E9 A9 8C 20 25 4A A9 A5 20
5010  25 4A 20 D0 50 8D F6 17 A5 EA 8D F5 17 A2 00 A1
5020  EA 48 A9 AA 81 EA 41 EA D0 21 68 81 EA A9 B6 20
5030  25 4A A9 00 8D FC 17 A9 50 8D FD 17 20 E6 50 30
5040  18 D0 11 A9 00 8D 00 2F 4C 0E 4B 68 A9 EF 20 25
5050  4A 4C 0D 50 A9 00 8D 00 2F 20 CA 42 A9 B6 20 C4
5060  42 20 ED 4A 20 FE 50 A5 EA 8D F7 17 A5 EB 8D F8
5070  17 C6 E9 10 09 A9 06 20 BB 42 A9 37 85 E9 4C 5C
5080  50 C9 8A F0 40 C9 9C D0 41 20 3F 49 C9 2C F0 2C
5090  C9 1E F0 12 C9 1A F0 19 C9 18 F0 23 C9 0E F0 22
50A0  29 77 F0 17 D0 24 20 D0 50 A9 E8 20 25 4A 20 FE
50B0  50 AD F7 17 85 EA AD F8 17 85 EB 60 4C 05 40 4C
50C0  0E 4B 4C DE 55 4C 35 54 30 B7 A9 C0 4C 17 53 EA
50D0  20 5F 53 10 FB A5 F6 29 7F C9 08 D0 F3 A5 F7 85
50E0  EA A5 F8 85 EB 60 20 3F 49 C9 0E F0 0E C9 02 F0
50F0  07 C9 3C D0 F1 A9 FF 60 A9 00 60 A9 0F 60 20 3F
5100  49 AA 8A 30 C3 85 F6 20 3F 49 85 F7 20 3F 49 85
5110  F8 20 3F 49 29 77 D0 B2 A9 03 85 E5 A9 43 85 E7
5120  A9 4E 85 E8 A9 00 20 81 53 30 3E A5 E4 C9 08 90
5130  3D C9 10 90 34 C9 20 90 3F C9 2C F0 2C C9 30 90
5140  41 A2 76 C9 34 90 5A A2 B6 C9 36 90 54 A2 36 C9
5150  38 90 4E C9 40 90 12 C9 48 90 4E C9 4A 90 54 C9
5160  4C 90 5C F0 66 C9 4D F0 69 A9 D8 4C 17 53 20 78
5170  53 E6 E4 A2 BE 4C DD 51 20 79 53 A9 08 A2 00 4C
5180  D9 51 4A 90 09 2A 20 79 53 A2 04 4C DD 51 C9 11
5190  D0 07 A9 14 A2 02 4C DB 51 20 79 53 A2 01 4C 3B
51A0  53 20 78 53 A9 02 4C D9 51 20 79 53 A9 8A A2 00
51B0  4C D9 51 29 01 20 78 53 A9 C0 A2 86 4C D9 51 29
51C0  01 20 78 53 A9 80 A2 B6 4C D9 51 A9 20 A2 06 4C
51D0  DB 51 A9 40 A2 01 4C DB 51 65 E4 86 EE 8A
51E0  D0 05 A2 01 4C 3B 53 20 3F 49 C9 8A D0 03 4C B7
51F0  53 C9 BC F0 4D C9 9E D0 0B A5 EE 29 09 F0 14 E6
5200  EE 20 3F 49 20 64 53 10 13 A5 F6 C9 1E D0 20 A5
5210  EE 29 40 F0 20 A9 08 A2 01 4C 2C 53 20 6C 53 10
5220  67 A5 F6 C9 98 F0 35 C9 A4 F0 6D 29 7F C9 08 D0
5230  6D A5 EE 29 04 F0 6D C5 EE D0 02 A9 00 A2 02 4C
5240  3B 53 20 5F 53 30 44 A5 E4 29 F0 C9 80 F0 04 A5
5250  EE 29 88 F0 0B 29 08 A2 02 4C 2C 53 A5 EE 29 20
5260  F0 42 20 3F 49 C9 2A D0 07 A9 14 A2 02 4C 2C 53
5270  C9 3A D0 0B 20 3F 49 C9 A4 F0 61 29 7F C9 08 D0
5280  1D A9 14 A2 02 4C 3B 53 20 6C 53 30 11 20 3F 49
5290  C9 98 F0 19 C9 A4 F0 6C F0 4F 29 7F C9 08 D0 20
52A0  A5 EE 29 03 F0 6B A9 0C A2 03 4C 3B 53 20 3F 49
52B0  85 F6 C9 2A D0 0C A5 EE 29 08 D0 19 A5 E4 C9 A2
52C0  D0 04 A5 F6 C9 3A D0 26 A5 EE 29 10 F0 2D A9 1C
52D0  A2 03 4C 2C 53 A9 18 A2 03 4C 2C 53 A5 EE C9 BF
52E0  D0 2F A9 00 A2 02 4C 2C 53 20 3F 49 C9 98 D0 05
52F0  20 3F 49 C9 2A D0 1E A5 EE 29 08 F0 18 A9 10 A2
5300  02 4C 2C 53 A5 EE C9 02 D0 0B A9 2C A2 03 4C 2C
5310  53 A9 E0 D0 0D A9 E0 48 20 3F 49 29 7F C9 08 D0
5320  F7 68 20 25 4A 60 EA EA EA EA EA EA 48 20 3F 49
5330  29 77 F0 06 68 A9 AC 4C 17 53 68 18 65 E4 85 F6
5340  A0 00 B9 F6 00 91 EA 51 EA F0 05 68 68 4C 4C 50
5350  C8 CA D0 EE 18 98 65 EA 85 EA 90 02 E6 EB 60 20
5360  3F 49 85 F6 20 1A 44 30 0E 20 26 44 20 3F 49 85
5370  F6 20 1A 44 20 26 44 60 0A 0A 0A 0A 0A 85 E4 18
5380  60 85 E6 A9 00 85 E4 A4 E5 88 B1 E6 C5 F8 D0 10
5390  88 B1 E6 C5 F7 D0 09 88 B1 E6 29 BE C5 F6 F0 12
53A0  E6 E4 18 A5 E6 65 E5 85 E6 A5 E4 C5 E8 90 D8 A9
53B0  FF 60 B1 E6 29 41 60 20 0A 55 10 2F 20 54 55 F0
53C0  04 A9 01 D0 02 A9 40 05 F6 85 F6 20 96 55 90 72
53D0  A9 FF 85 F7 85 F8 20 54 55 F0 06 A9 0C A2 03 D0
53E0  04 A9 00 A2 02 20 3B 53 60 00 00 48 20 54 55 F0
53F0  1E 68 C9 40 F0 1C 6A A0 01 B1 E6 85 F8 88 B1 E6
5400  85 F7 90 D7 CB B9 EA 00 91 E6 88 10 F8 30 CC 68
5410  C9 01 F0 17 2A 2A 20 6A 55 D0 15 85 F7 90 C2 A0
5420  01 B9 EA 00 91 E6 88 10 F8 30 B6 A9 B8 4C 17 53
5430  A9 D0 4C 17 53 20 0A 55 10 12 20 96 55 90 03 4C
5440  FE 50 A9 B8 20 25 4A A9 CB 4C 22 53 D0 05 A9 B8
5450  4C 22 53 C9 40 F0 42 A5 E6 48 A5 E7 48 A0 02 B1
5460  E6 29 BE 91 E6 88 B1 E6 48 88 B1 E6 85 E6 68 85
5470  E7 A0 02 B1 E6 48 88 B1 E6 48 88 B9 EA 00 C8 91
5480  E6 B9 EA 00 C8 91 E6 68 85 E6 68 85 E7 C9 FF D0
5490  E0 A5 E6 C9 FF D0 DA F0 3C A5 E6 48 A5 E7 48 A0
54A0  02 B1 E6 29 BE 91 E6 88 B1 E6 48 88 B1 E6 85 E6
54B0  68 85 E7 20 B7 55 D0 31 85 E4 A0 01 B1 E6 48 A5
54C0  E4 91 E6 68 C9 FF F0 0D 18 69 02 65 E6 85 E6 B0
54D0  E2 C6 E7 90 DE 68 85 E7 68 85 E6 A0 00 A5 EA 91
54E0  E6 C8 A5 EB 91 E6 4C FE 50 68 68 A9 B6 20 C4 42
54F0  A5 E6 D0 02 C6 E7 C6 E6 A5 E7 20 C7 42 A5 E6 20
5500  C7 42 20 BE 42 A9 D0 4C 22 53 20 3F 49 85 F6 20
5510  3F 49 85 F7 29 77 F0 1A 20 3F 49 85 F8 29 77 F0
5520  13 20 3F 49 29 7F C9 08 F0 0C A9 C0 20 25 4A 68
5530  68 60 85 F7 85 F8 A5 E4 85 ED A9 05 85 E5 A9 2F
5540  85 E7 A9 01 AE 00 2F 86 E8 F0 04 20 81 53 60 85
5550  E6 A9 FF 60 A5 ED 85 E4 A5 EE F0 07 C9 04 F0 02
5560  B0 01 60 68 68 A9 E0 4C 22 53 08 A0 00 38 B1 E6
5570  F9 EA 00 08 3B E9 02 28 48 C8 B1 E6 F9 EA 00 D0
5580  05 68 30 0E 10 07 C9 FF D0 07 68 10 05 28 A0 00
5590  60 68 28 A0 FF 60 A9 32 CD 00 2F 90 19 A0 02 B9
55A0  F4 00 91 E6 C8 C4 E5 D0 F6 A0 01 B9 EA 00 91 E6
55B0  88 10 F8 EE 00 2F 60 08 38 A5 EA E5 E6 08 38 E9
55C0  02 28 48 A5 EB E5 E7 D0 05 68 30 0E 10 07 C9 FF
55D0  D0 07 68 10 05 28 A0 00 60 68 28 A0 FF 60 20 CA
55E0  42 AD 00 2F 85 E8 F0 36 A9 2F 85 E5 A9 01 85 E4
55F0  A0 02 B1 E4 29 41 F0 18 A0 00 B1 E4 85 EA C8 B1
5600  E4 85 EB 20 ED 4A C8 B1 E4 20 C4 42 C0 04 D0 F6
5610  18 A9 05 65 E4 85 E4 C6 E8 D0 D5 20 CA 42 A9 2F
5620  85 EB A9 00 85 EA 20 ED 4A AD 00 2F 20 C7 42 68
5630  68 4C 00 50
```

Bild 4. Das Assemblerprogramm als Hex-Dump. Es umfaßt 1,5 KByte und läßt sich in einem EPROM unterbringen

```
SPEICHERTABELLE VON 4000  BIS 4400
4000
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
4000  A2 06 20 4C 47 A9 35 85 E9 D8 A9 2F 85 ED A9 4F
4010  85 E3 20 63 42 C6 E9 10 09 A9 06 20 BB 42 A9 37
4020  85 E9 20 47 40 18 A5 E8 65 EA 85 EA A9 00 65 EB
4030  85 EB 38 AD F7 17 E5 EA AD F8 17 E5 EB B0 D6 A9
4040  03 20 BB 42 6C FC 17 20 CA 42 A5 EB 20 C7 42 A5
4050  EA 20 C7 42 A9 03 20 C1 42 20 DF 41 A9 02 18 65
4060  E7 20 C1 42 A9 01 85 EC 85 E6 20 6E 40 60 A0 00
4070  B1 EA 85 EE C9 89 F0 31 C9 80 F0 2D C9 A2 D0 0B
4080  A0 9F 20 CB 41 A0 1E 20 CB 41 60 29 FD C9 9C F0
4090  18 A5 EE 29 DF C9 DA F0 10 29 9F C9 1A F0 0A 29
40A0  0F C9 02 F0 04 29 03 C9 03 F0 7B C9 01 D0 0B A2
40B0  00 20 A7 41 A2 04 20 A7 41 60 A5 EE 29 0F C9 08
40C0  D0 06 A2 08 20 A7 41 60 C9 00 D0 38 A5 EE 29 9F
40D0  C9 80 F0 42 A2 0C 20 A7 41 98 29 01 F0 1A A8 B1
40E0  EA 48 10 04 A9 B4 D0 02 A9 9A 20 C4 42 18 68 65
40F0  EA 18 69 02 20 C7 42 60 A5 EE C9 20 90 05 D0 45
4100  20 03 42 60 C9 0A D0 10 A5 EE 29 8F C9 8A D0 08
4110  A2 10 20 A7 41 60 F0 52 A5 EE 29 F7 C9 24 D0 08
4120  A0 E4 20 CB 41 60 F0 74 A5 EE 29 DF C9 4C D0 1B
4130  A0 E7 20 CB 41 A5 EE C9 6C F0 05 20 03 42 F0 05
4140  A0 84 20 CB 41 C6 E9 20 CA 42 60 29 03 C9 02 D0
4150  19 A2 14 20 A7 41 A5 EE 29 D7 C9 96 F0 06 A2 18
4160  20 A7 41 60 A2 1C 20 A7 41 60 A5 EE 29 D3 C9 C0
4170  D0 0B A2 20 20 A7 41 A2 24 20 A7 41 60 29 C3 C9
4180  80 D0 19 A2 28 20 A7 41 A0 00 B1 EA C9 A0 F0 06
4190  A2 2C 20 A7 41 60 A0 1E 20 CB 41 60 A0 EA 20 CB
41A0  41 A0 ED 20 CB 41 60 A0 00 B1 EA 3D D0 42 85 F7
41B0  E8 BD D0 42 E8 C5 F7 F0 06 7D D0 42 C8 90 F6 84
41C0  F7 06 F7 98 65 F7 E8 7D D0 42 A8 A2 03 B9 00 43
41D0  20 C4 42 CA F0 03 C8 D0 F4 E6 E6 20 BE 42 60 A9
41E0  03 85 E7 A0 00 B1 EA 20 2B 42 85 E8 AA B1 EA 20
41F0  C7 42 20 BE 42 C8 C6 E7 CA D0 F2 A5 E7 0A 65 E7
4200  85 E7 60 AD 00 2F F0 22 D0 09 18 A9 05 65 EC F0
4210  19 85 EC A0 00 B1 EC C8 D1 EA D0 EE C0 01 F0 F5
4220  B1 EC 20 C4 42 C8 C0 05 D0 F6 60 C9 20 F0 31 C9
4230  A2 F0 26 29 9F C9 00 F0 24 29 1F C9 19 F0 21 29
4240  0F C9 02 F0 18 48 29 03 C9 03 F0 10 68 29 0D C9
4250  08 F0 0A 29 0C C9 0C F0 07 A9 02 60 68 A9 01 60
4260  A9 03 60 A9 03 20 BB 42 A9 7D 20 25 4A A2 00 A0
4270  CF 20 B3 42 EB E8 A0 D5 20 B3 42 20 CA 42 A9 2F
4280  85 ED 60 20 CD 42 BD F6 17 95 EB 20 C7 42 BD F5
4290  17 95 EA 20 C7 42 20 BE 42 60 85 FC 68 18 69 29
42A0  8D 81 17 68 8A 48 98 48 08 A5 FC 4C 80 17 28 68
42B0  A8 68 AA A5 FC 60 FF FF 20 9A 42 20 9A 42 20 9A
42C0  42 20 9A 42 20 9A 42 20 9A 42 20 9A 42 20 9A 42
42D0  E0 00 20 00 1F 01 04 18 F0 00 10 30 F0 00 10 60
42E0  F0 80 10 C0 E0 00 20 90 1F 06 04 AB 1F 16 04 B4
42F0  F0 C0 20 D8 1F 00 02 78 E0 80 20 DE 1F 04 04 AB
4300  30 14 06 06 18 12 02 30 14 06 12 1C 0A 20 06 24
4310  12 06 1C 38 2C 0A 32 1C 98 3A A4 08 08 08 0C 38
4320  08 08 08 08 A4 98 2A 22 98 3A 06 98 2A 06 98 3A
4330  2C 28 2C 1C 24 1C 2C 24 2C 0A 02 1C 2C 28 06 1C
4340  24 0C 2C 24 06 0A 02 0C 12 02 2A 20 2A 06 20 06
4350  2A 1C 24 3C 0C 18 2A 1C 24 12 0C 18 3A 0A 02 12
4360  32 14 1E 32 2C 24 16 0A 14 32 38 0C 14 20 0C 32
4370  3C 1C 14 20 0A 32 3C 0A 0C 38 08 32 1C 1C 08 08
4380  08 32 1C 0A 0C 18 12 32 18 02 08 08 08 32 02 2E
4390  06 0A 24 14 30 24 24 0A 14 14 30 14 0A 20 3A 24
43A0  12 3A 12 02 1C 0C 18 1C 08 08 08 08 08 08 06 1E
43B0  08 08 08 08 22 98 2A 22 98 3A 06 98 2A 06 98 3A
43C0  20 3A 06 20 3A 0A 20 06 3A 20 0A 3A 12 02 3A 3C
43D0  30 18 18 30 2C 32 0C 0A 1C 2C 2A 1C 2C 3A 0A 20
43E0  2A 24 12 2A 32 0C 20 16 38 2C 96 18 30 30 2C 1C
43F0  3E 2C 14 30 34 14 06 38 38 24 0C 0A 20 02 08 7E
```

Bild 5. Das mit dem Assembler zusammenarbeitende Disassembler-Programm. Es enthält auch die Sprünge zu den Ausgaberoutinen

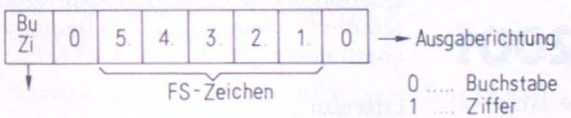


Bild 6. Codierung der seriellen Fernschreibzeichen im Speicher

```
SPEICHERTABELLE VON 2000  BIS 2199
2000
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
2000  48 A9 7F 2D 00 17 8D 00 17 20 5D 20 A2 05 68 4A
2010  4A 48 AD 00 17 90 04 09 80 D0 02 29 7F 8D 00 17
2020  20 5D 20 CA D0 EA 09 80 8D 00 17 68 20 5D 20 20
2030  72 20 60 2C 00 17 50 22 2C 00 17 70 FB 20 72 20
2040  20 5D 20 A2 05 2C 00 17 50 03 38 B0 01 18 6A 20
2050  5D 20 CA D0 F0 4A 4A 4A 0A 60 A9 FF 60 A0 13 8C
2060  07 17 2C 05 17 10 FB A0 44 8C 05 17 2C 05 17 10
2070  FB 60 A9 09 4C 5F 20 29 FE 48 10 10 A5 E1 29 20
2080  D0 1B A5 E1 09 20 85 E1 A9 36 D0 0E A5 E1 29 20
2090  F0 0B A5 E1 29 DF 85 E1 A9 3E 20 00 20 68 20 00
20A0  20 60 20 33 20 20 B8 20 C9 3E F0 F6 C9 B6 F0 F2
20B0  C9 96 D0 03 6C FC 17 60 48 C9 3E F0 0A C9 36 D0
20C0  0C A5 E1 09 02 D0 04 A5 E1 29 FD 85 E1 A5 E1 29
20D0  02 D0 02 68 60 68 09 80 60 29 BE 0A 90 02 09 80
20E0  4A 4A AA BD 00 22 48 A9 7F 2D 00 17 8D 00 17 20
20F0  60 21 A2 08 68 4A 48 AD 00 17 90 04 09 80 D0 02
2100  29 FE 8D 00 17 20 60 21 CA D0 E9 09 80 8D 00 17
2110  68 20 60 21 60 20 38 21 A2 3F DD 00 22 F0 05 CA
2120  10 F8 30 F1 8A 0A 85 DC 24 DC 50 02 09 80 29 BF
2130  C9 96 D0 03 6C FC 17 60 2C 00 17 50 20 2C 00 17
2140  70 FB 20 6E 21 20 60 21 A2 08 2C 00 17 50 03 38
2150  B0 01 18 6A 20 60 21 CA D0 F0 0A 4A 60 A9 FF 60
2160  A0 95 EA EA EA EA EA EA EA EA D0 F5 60 A0 4A
2170  4C 62 21 A9 08 20 C4 42 60 A9 10 20 C4 42 A9 04
2180  20 C4 42 A9 10 20 C4 42 60 AA 20 73 21 CA D0 FA
2190  60 AA 20 7E 21 CA D0 FA 60
SPEICHERTABELLE VON 2200  BIS 2240
2200
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
2200  00 45 0A 41 20 53 49 55 0D 44 52 4A 4E 46 43 4B
2210  54 5A 4C 57 48 59 50 51 4F 42 47 00 4D 58 56 00
2220  00 33 0A 2D 20 60 38 37 0D 23 34 07 2C 2B 3A 28
2230  35 2B 29 32 5F 36 30 31 39 3F 2D 00 2E 2F 3D 00
```

Bild 7. Hex-Dump der Ein/Ausgaberoutinen. Im Adressenbereich 2200...223F befindet sich die ASCII-Baudot-Umwandlungstabelle

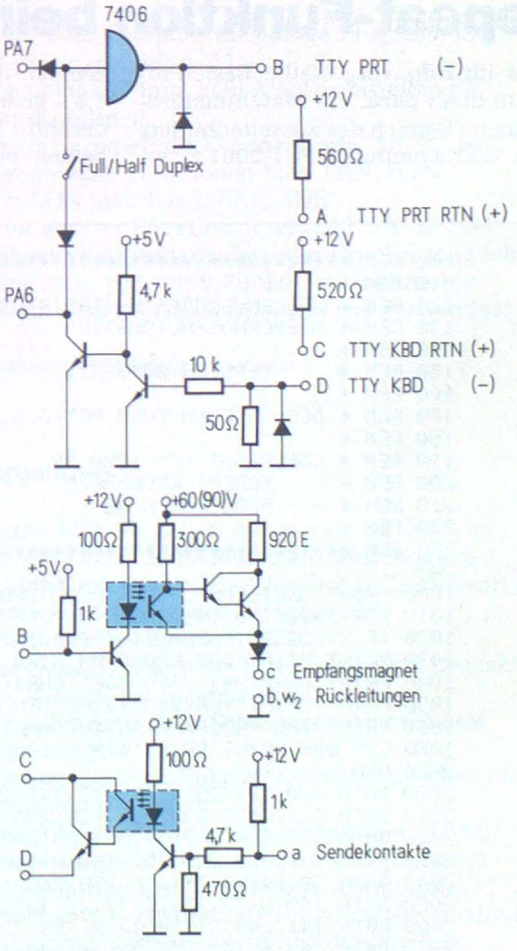


Bild 8. Hardware zum wahlweisen Anschluß von Baudot oder ASCII-Fernschreibern bzw. -Terminals

XX E6 JMP NZW n (= Akkuinhalt) mal Zeilenvorschub und Wagenrücklauf
XX E9 JMP 1SP 1mal Spaltensprung (Zwischenraum)
XX EC JMP NSP n mal Spaltensprung (Zwischenraum)
XX EF JMP FZA Fernschreibzeichen (im Akku) wird auf Fernschreiber ausgegeben
XX F2 JMP DRH drucke Akkuinhalt als 2 Hexadezimalzahlen
XX F5 JMP WZW Wagenrücklauf, Zeilenvorschub
XX F8 4C CB 41 intern notwendiger Sprung

Der Einsprung in das FS-Zeichen Einleseprogramm FZE erfolgt direkt auf 493F bzw. bei XXE3. *Bild 6* zeigt die Codierung der Baudot-Fernschreibzeichen im Speicher, speziell an den Programmadressen 46D2 bis 4FFF. Das Unterprogramm an der Adresse 441A wandelt den vom Fernschreiber empfangenen Baudot-Code in eine Hex-Zahl. Die Ausgabe von Text erfolgt mit dem Unterprogramm bei 4817; eine Hex-Adresse wird mit der Subroutine bei 4ADF ausgegeben.

Der Vollständigkeit halber sind die von Walter Werner entwickelten Ein-/Ausgaberoutinen in *Bild 7* wiedergegeben; im Adressenbereich 2200...223F befindet sich die ASCII-Baudot-Umwandlungstabelle. Die für den Anschluß des Fernschreibers nötige Hardware ist schließlich in *Bild 8* wiedergegeben. Die Verwendung von Optokopplern empfiehlt sich unbedingt wegen der sauberen galvanischen Trennung von Baudot-Fernschreibern. Im ASCII-Ein-/Ausgabeteil von Bild 8 wurde der Einfachheit halber hierauf verzichtet.

Automatische Zeilennumerierung und Repeat-Funktion beim PET 2001

Das im *Bild* vorgestellte Basic-Programm dient dazu, ein Maschinenprogramm im Bereich des Kassettenbuffers 2 des Mikrocomputers PET-2001 zu erzeugen, das jede Taste, die länger als 1,5 s gedrückt wird, automatisch wiederholt. Ferner ist das automatische Setzen von Zeilennummern mit der Schrittweite 10 möglich. Die Schrittweite kann durch POKE 883,XX variiert werden.

Das Basic-Ladeprogramm ist selbsterklärend. Wird die Meldung DATA ERROR ausgegeben, so ist ein Fehler beim Eintippen der DATA-Statements gemacht worden. Da ROM-Routinen und Zero-Page-Adressen benutzt werden, ist das Programm so nur auf dem „alten" PET-Betriebssystem lauffähig, nicht auf dem CBM-System. Angaben zur Änderung der Adressen finden sich in [1].

Automatische Zeilennummern werden nur dann gesetzt, wenn sich die Länge des Basic-Programms durch Eingabe einer neuen Programmzeile ändert. Daraus ergibt sich, daß nach Eingabe von NEW und Neueingabe einer Zeile, die ebenso lang ist wie das soeben gelöschte Programm, keine neue Zeilennummer ausgegeben wird. Doch wird dies wohl ein Ausnahmefall bleiben.

Gelegentlich kommt es vor, daß beim Editieren nach dem Drücken der Return-Taste ein Quadrat dort stehen bleibt, wo der Cursor stand. Dieses Quadrat wird vom Rechner jedoch ignoriert, wenn eine Programmzeile übertragen wird.

Robert Kröpelin

Literatur

[1] ROM und RAM in PET und CBM. FUNKSCHAU-Sonderheft „Mikrocomputer-Anwendungen" (Franzis-Verlag)

Dieses Basic-Initialisierungsprogramm erzeugt ein Maschinensprache-Programm, das automatisch Basic-Zeilennummern erzeugt und außerdem eine Repeat-Funktion aller Tasten simuliert

```
100 REM ********************************
110 REM *                              *
120 REM * ZEILENNUMMER- U. TASTATUR-   *
130 REM * WIEDERHOL-AUTOMATIK          *
140 REM *                              *
150 REM *      VERSION 80/09/11.1      *
160 REM *                              *
170 REM * GESCHRIEBEN FUER PET-O.S.    *
180 REM *                              *
190 REM * COPYRIGHT (C) 1980 BY        *
200 REM *      ROBERT KROEPELIN        *
210 REM *      5000 KOELN 60           *
220 REM *                              *
230 REM ********************************
240 :
1000 PRINT"ZEILENNR.- U. TASTATURWIEDERHOLAUTOMATIK"
1010 FOR I=826 TO 1012:READ A:POKE I,A:X=X+A:NEXT
1020 IF X<>22595 THEN PRINT"### DATA ERROR ###":END
1030 PRINT:PRINT"DAS PROGRAMM WIRD JEWEILS MIT DEM BEFEHL"
1040 PRINT CHR$(34);"SYS 826";CHR$(34);" EIN- UND AUSGESCHALTET."
1050 PRINT:PRINT"VOR BENUTZUNG DES KASSETTENRECORDERS IST"
1060 PRINT"DAS PROGRAMM AUSZUSCHALTEN."
1070 SYS 826:PRINT:PRINT"*** AUTOMATIK JETZT AKTIV ***"
1080 NEW
2000 DATA 120,173,25,2,73,213,141,25,2,173,26,2,73,229,141,26,2,32,146,3,88,96
2010 DATA 32,160,3,234,165,124,205,245,3,208,7,165,125,205,246,3,240,71,169,4
2020 DATA 197,124,208,4,197,125,240,61,206,247,3,208,30,24,169,10,101,8,170,164
2030 DATA 9,144,1,200,152,160,0,140,39,2,132,226,32,159,220,169,32,32,210,255
2040 DATA 32,146,3,76,126,230,165,124,141,245,3,165,125,141,246,3,141,247,3,96
2050 DATA 169,0,72,72,72,72,76,133,230,173,3,2,201,255,240,64,205,248,3,240,28
2060 DATA 141,248,3,174,13,2,208,2,162,10,189,14,2,141,249,3,169,60,141,250,3
2070 DATA 169,6,141,251,3,240,190,173,250,3,240,5,206,250,3,208,180,173,251,3
2080 DATA 240,5,206,251,3,208,170,169,6,141,251,3,173,249,3,76,44,231,141,248,3
2090 DATA 240,154
READY.
```

Tischcomputer-Auswahlkriterien

Die Hersteller heutiger Tischcomputer hatten bei der Entwicklung meist einen ganz bestimmten Verwendungszweck im Auge – „den" universellen Computer gibt es nicht. Es bleibt also dem Anwender überlassen, die richtige Auswahl für seine individuellen Anforderungen zu treffen. Die nachfolgend wiedergegebenen Auswahlkriterien für typische Aufgabenstellungen sollen dabei behilflich sein.

Die vier Hauptanwendungsgebiete von Tischcomputern sind die Steuerung von externen Geräten und Anlagen z. B. einer Werkzeugmaschine, einer Heizung o. ä.; die Meßtechnik, wobei ein Computer ein oder mehrere Meßgeräte abfragt und die Meßergebnisse auswertet sowie protokolliert; die Mathematik, zu der auch die statistische Auswertung von Daten oder etwa die Dimensionierung elektronischer Schaltungen gehört; und schließlich die Textverarbeitung, die das Schreiben und Redigieren von Texten, Briefen, Adressenkarteien usw. umfaßt.

Es lohnt sich, für unterschiedliche Tischcomputer des derzeitigen Angebots eine Strichliste zu führen, wieviele Forderungen für den individuellen Verwendungszweck erfüllt und wie viele nicht erfüllt werden können. Dabei sollte man möglichst gerecht vergleichen und berücksichtigen, welche Forderungen nur mit u. U. recht teuren Zusatzeinrichtungen erfüllt werden können. Entscheidend ist letztlich der Preis für diejenige Systemkonfiguration (Computer, Drucker, Massenspeicher, sonstige Zusatzgeräte), die für die Aufgabenstellung erforderlich ist. Fe.

Steuerungsaufgaben in Labor, Fertigung und Prüffeld

- ☐ Mindestens 1 KByte freier Arbeitsspeicher (RAM)
- ☐ Assembler/Maschinensprache-Programmierung bequem möglich, ausreichende Systemdokumentation hierfür (Adressenbelegung usw.)
- ☐ Verbreiteter Prozessortyp (8080, 8085, Z-80, 6502) wegen Software-Unterstützung
- ☐ Mindestens 16 freie I/O-Leitungen (2 Ports)
- ☐ Prozessor-Reset ohne Programmverlust möglich, z. B. bei „aufgehängten" Maschinensprache-Programmen
- ☐ Externe Programmspeicherung preiswert möglich, z. B. mit Kassettenrecorder
  Interrupt-Timer eingebaut und vom Benutzer programmierbar

Meßtechnik

- ☐ IEC-Bus-Anschluß, ev. durch Software simulierbar
- ☐ I/O-Möglichkeiten (Parallel- und/oder Seriell-Port)
- ☐ Interrupt-Timer eingebaut und frei programmierbar
- ☐ Abruf der Uhrzeit möglich
- ☐ Assembler-/Maschinensprache-Programmierung möglich bzw. z. B. vom Basic-Interpreter aus mit PEEK/POKE/USR möglich
- ☐ Höhere Programmiersprache, z. B. Basic, mit speziellen Befehlen für IEC-Bus- und I/O-Programmierung (IN/OUT bzw. PEEK/POKE; OPEN/CLOSE)
- ☐ Verbreiteter Prozessortyp (8080, 8085, Z-80, 6502) wegen Software-Unterstützung
- ☐ Ausreichende Dokumentation des Systems, wie Adressenbelegung von ROM und RAM
- ☐ Mindestens 4 KByte freier Arbeitsspeicher (RAM)

Mathematik, Statistik, Entwicklung

- ☐ Interpreter oder Compiler für höhere Programmiersprache (Basic, Pascal, o. ä.)
- ☐ Rechengenauigkeit min. acht Mantissenstellen im Fließkomma-Format
- ☐ Fließkomma-Format mit Exponent min. $10^{-30}...10^{+30}$
- ☐ Trigonometrische Funktionen SIN, TAN, ATN
- ☐ Funktion LOG (nat. Log.), EXP, SQR
- ☐ Definition eigener Funktionen mit DEF FN möglich
- ☐ Kurvendarstellung auf Display oder Drucker möglich, Auflösung min. 100 x 100 Punkte, ev. mehrfarbig
- ☐ Mindestens 4 KByte Arbeitsspeicher frei verwendbar (RAM)
- ☐ Getrenntes numerisches Tastenfeld

Textverarbeitung

- ☐ Bildschirm: Min. 16 Zeilen mit je min. 64 Zeichen
- ☐ Anzeige von Groß- und Kleinbuchstaben möglich
- ☐ Tastatur für Groß- und Kleinschreibung: Großschrift bei gedrückter Shift-Taste, nicht umgekehrt!
- ☐ Deutsche Sonderzeichen ä, ö, ü, ß
- ☐ Deutsche Sonderzeichen auch bei passenden Druckern möglich, speziell bei Schönschreib-Druckern
- ☐ Tastaturausführung nach Schreibmaschinen-Norm (y/z vertauscht?)
- ☐ Cursorbewegung nach links zu Korrekturzwecken ohne Zeichenlöschung möglich
- ☐ Mindestens 16 KByte freier Arbeitsspeicher (RAM)
- ☐ Textverarbeitungs-Software existent und preiswert erhältlich (Mailing List = Adressenverwaltung, Word Processor = Textformatierung und -bearbeitung)
- ☐ Höhere Programmiersprache mit String-Befehlen (in Basic: MID$, LEFT$, RIGHT$, LEN usw.)
- ☐ Mehrere Floppy-Disk-Laufwerke gleichzeitig anschließbar

Allgemeine Bewertung von Tischcomputern

- ☐ Ist die verwendete Programmiersprache sofort beim Einschalten verfügbar (ROM-resident)? Wenn nein: Wie lange dauert der Ladevorgang, und ist er einfach zu handhaben?
- ☐ Welcher Teil der Arbeitsspeicher-Kapazität steht dem Anwender beim Arbeiten mit einer höheren Programmiersprache (meist Basic) tatsächlich zur Verfügung?
- ☐ Ist Zubehör (Drucker usw.) nur vom Computerhersteller selbst zu haben, oder gibt es auch Fremdfabrikate zur Auswahl?
- ☐ Ist das Datenaufzeichnungsformat von Kassetten oder Disketten mit anderen Geräten kompatibel (z. B. IBM-Format)?
- ☐ Stehen unterschiedliche Drucker zur Verfügung (breit, schmal, Matrixdrucker, Typenraddrucker usw.)?
- ☐ Ist bei Ausfällen ein schneller Service sichergestellt, z. B. durch ein Austauschgerät vom regionalen Kundendienst?
- ☐ Ist in Ihrer Nähe eine Kundendienststelle der Herstellerfirma?
- ☐ Wird Standardsoftware beim Computerkauf mitgeliefert?
- ☐ Kennen Sie jemanden, der den gleichen Computertyp verwendet?
- ☐ Haben Sie ein Programm für diesen Computer schon einmal in einer Zeitschrift gesehen?
- ☐ Kennen Sie ein Software-Büro, das Programme für diesen Computertyp erstellen kann?
- ☐ Können Sie den Computer nach dem Auspacken sofort selbst in Betrieb nehmen (220-V-Anschluß, Komplettsystem)?
- ☐ Falls der Computer keinen eingebauten Bildschirm, sondern einen Fernsehgeräte-Ausgang besitzt: Kann ein Fernsehgerät deutscher Norm (50 Hz Bildwechselfrequenz, PAL-Farbfernsehnorm, 625 Zeilen) ohne Umrüstung verwendet werden?
- ☐ Sind die Handbücher in deutscher Sprache geschrieben?
- ☐ Liefert der Computer Fehlermeldungen im Klartext oder nur als Fehler-Codenummer?
- ☐ Haben Sie von der mechanischen Ausführung (Gehäuse, Tastatur) einen guten Eindruck?
- ☐ Stehen die Kosten des Gesamtsystems in einem guten Verhältnis zu den Einsparungen, die Sie sich davon erwarten?

Bildschirm oder einzeiliges Display?

Anzeigentyp	Vorteile	Nachteile
Bildschirm	Übersichtliche Darstellung von Tabellen, Texten, Programmen	Einschaltverzögerung, Stromverbrauch, Größe, Gewicht, streufeldempfindlich, teuer
Siebensegm.-Display	Wenig Stromverbrauch, für Steuerungs-Computer ausreichend, klein	Nur für hexadezimale Maschinenprogrammierung geeignet
Alphanumerisches Display	Wenig Stromverbrauch, klein, preiswert, für Assembler- und einfache Basic-Programme ausreichend	Nur eine Zeile mit z. B. 20 Zeichen darstellbar, für Textverarbeitung ungeeignet

Auswahl des Druckers

Druckertyp	Vorteile	Nachteile
Thermodrucker (Matrix)	Recht leise, preiswert, schnell, wenig mechanisch bewegte Teile	Teures Thermopapier; blaues Papier nicht kopierfähig; nicht für Durchschläge
Metallpapier (Matrix)	Recht leise, preiswert, schnell, zuverlässig	Teures Metallpapier, empfindlich gegen Fingerabdrücke; schlecht lesbar, meist geringe Breite, nicht für Durchschläge
Normalpapier-Matrixdrucker	Schnell, gute Druckqualität, große Papierbreite möglich, Durchschläge möglich	Relativ teures Druckwerk, ziemlich laut, meist nur für gelochtes EDV-Papier
Typenhebel-Normalpapier-Drucker (oder Typenrad)	Sehr gute Schriftqualität (für Briefe geeignet), normale DIN-Papierbögen verwendbar	Teures Druckwerk, relativ langsam

Anmerkung: Manche Matrixdrucker (z. B. CBM 3022 oder die in PC-100, AIM-65 und HP-85 eingebauten Thermodrucker) eignen sich zum Plotten mit der Auflösung eines Matrixpunktes, da jeder Punkt per Software getrennt ansteuerbar ist.

Computer an Schulen

Die einzelnen Länder der Bundesrepublik haben im Augenblick noch ziemlich unterschiedliche Vorstellungen, was den Einsatz von programmierbaren Rechnern und Computern an Schulen angeht. Die Situation in Bayern sieht im Moment so aus:

Im „Amtsblatt des Bayerischen Staatsministeriums für Unterricht und Kultus", B1234 A, Sonder-Nr. 28, wird der Lehrplan für das Fach Informatik an den 10. Klassen der bayerischen Gymnasien festgelegt, wie es für den mathematisch-naturwissenschaftlichen Zweig vorgesehen ist. Vermittelt werden sollen danach die Kenntnis des Begriffs „Algorithmus" als eindeutiges Lösungsverfahren sowie der Steueranweisungen „Auswahl" und „Wiederholung"; Einblicke in den Variablenbegriff; die Fähigkeit, einfache Programme zu lesen, zu durchschauen und auch zu schreiben; unterschiedliche Datentypen und Dateiarten kennenzulernen und – nicht zu vergessen – ein Einblick in die Geschichte der Datenverarbeitung sowie in ihre gesellschaftlichen Auswirkungen und Zusammenhänge. Im Klartext: „Offenheit gegenüber den sozialen und wirtschaftlichen Konsequenzen, die mit der Zunahme der EDV verbunden sind."

Während ursprünglich geplant war, jeder Schule, die diesen Informatikunterricht durchführt, acht Tischcomputer für Lehrzwecke zur Verfügung zu stellen, gehen die derzeitigen Pläne von nur noch vier Computern aus – drei CBM 3008 und ein CBM 8032. Fe.

Werner Frangen

Ein Stack für komplexe Zahlen

Schön wär's, mit komplexen Zahlen ebenso im Stack rechnen zu können, wie (bei HP-UPN-Rechnern) mit reellen Zahlen. Ein „komplexer Stack" kann aber simuliert werden. Unser Programm beschränkt sich auf drei Stack-Ebenen, um Zeit und Speicherplatz zu sparen. Außerdem werden bis zu sieben komplexe Zahlen in Registerpaaren gespeichert. Zwei Tricks kommen zur Anwendung, die näher erläutert werden: die indirekte Adressierung mit dem i-Register zur Vermeidung von zeitraubenden Datentransporten[1]), und die Mehrfachausnützung von Labels bei HP-67 und HP-97.

Zum Unterschied vom Zeiger i wird die komplexe Einheit, wie in der Technik üblich mit $j = \sqrt{-1}$ bezeichnet. Der simulierte „Stack" und die hierauf bezüglichen Operationen wie „Enter" werden in Anführungszeichen gesetzt, um sie vom Rechner-Stack zu unterscheiden. Das HP-67-Programm zeigt *Tabelle 1*.

Der „Stack" beschränkt sich auf drei „Levels" für je eine komplexe Zahl:

L0 enthält $z = x + yj$ (x im X-, y im Y-Register);
L1 enthält $w = u + vj$ (u in R_i, v in RS_i);
L2 enthält $t = p + qj$ (p in R_k, q in RS_k).

Dabei ist i = 0 oder 1 der jeweilige Zeigerstand und k = 1 oder 0 der von i verschiedene Wert. Mit R_n werden die Primär-, mit RS_n die Sekundärregister des HP-67/97 bezeichnet (n = 0, ..., 9). Außerdem sind sieben fest adressierbare Registerpaare für komplexe Zahlen vorgesehen:

C_m enthält $c_m = a_m + b_m j$ (a_m in R_m, b_m in RS_m; m = 3, ..., 9).

Bei der Eingabe einer komplexen Zahl in L0 geht der Imaginärteil voran! Angezeigt wird immer der Realteil x. Mit den Tasten A R/S wird der Imaginärteil y in einer Pause angezeigt. Nur im Einzelschritt-Verfahren – SST – ist dieser Weg nicht gangbar, weil man damit den betr. Programmschritt verläßt. Bringt man y mit der Taste x⇔y zur Anzeige, so muß man anschließend erneut x⇔y drücken, um die richtige Reihenfolge wiederherzustellen!

Über die Befehlsmöglichkeiten und ihre Wirkung auf die jeweiligen „Stack"-Inhalte z, w, t unterrichtet *Tabelle 2*. Nur in drei Kleinigkeiten gibt es eine Abweichung vom gewohnten Stack-Verhalten:

1. Die Eingabe einer komplexen Zahl in L0 ist natürlich nicht mit einem automatischen „Stacklift" verbunden. Wird ein entsprechender „Stacklift" gewünscht, so muß man vor der Zahleingabe „Enter" (e) drücken.

2. Dagegen ist ein „Recall" (m D) stets mit einem „Stacklift" verbunden, auch wenn ein „Enter" unmittelbar voranging. Zum Ausgleich dafür ist auch ein Befehl „CLz&Recall" (m C) ohne „Stacklift" vorgesehen. Er „überschreibt" (oder verdrängt) den vorherigen L0-Inhalt z ebenso, wie es jede komplexe Zahl-Eingabe tut. Ein besonderer „CLz"-Befehl (Lösche z) wird dadurch überflüssig.

3. Ein besonderes „LAST-z-Register" ist nicht vorgesehen. Soll der Operand z vor Ausführung einer Rechenoperation „gerettet" werden, so kann dies der Benutzer mittels „Store" (m B) tun.

Wohl aber wird bei zweistelligen Rechenoperationen die Zahl t aus L2 nach L1 hinein „dupliziert".[2])

[1]) Nach einer Idee von Peter Van Den Hamer; siehe [1].

[2]) Diese für die rationelle Polynomberechnung nach dem Horner-Verfahren wichtige Eigenschaft fehlt dem Programm von Van Den Hamer in [1].

Simulation des Stackverhaltens

Um Zeit zu sparen, wird die Duplizierung nicht ausgeführt; es wird lediglich der Zeiger i „umgeschaltet" (0⇔1) und ein Flag F0 gesetzt. Die Umschaltung des Zeigers simuliert die Vertauschung der Inhalte von L1 und L2 (L1⇔L2). Das gesetzte F0 soll bedeuten, daß der Inhalt von L2 gleich dem von L1 angenommen wird (obwohl das in Wirklichkeit meist nicht der Fall ist). L2 ist gleichsam „vorübergehend stillgelegt".

Um Zeiger und Flag F0 braucht sich der Benutzer indessen gar nicht zu kümmern. Er arbeitet mit dem „Stack" so, wie es in der Benutzeranweisung erklärt ist. – Lediglich bei Error-Anzeige infolge Fehlbedienung empfiehlt es sich, mit x⇔I nachzusehen, ob ein falsches i (≠0 oder 1) dazu Veranlassung gegeben hat. Nötigenfalls wird die Lage durch eine Gesamtlöschung (nach Anweisung Nr. 2) bereinigt.

Als vielseitig anwendbare Programmiertechnik verdient aber die „Stack"-Simulation doch eine nähere Erläuterung.

L1⇔L2 wird durch die Zeigerumschaltung (Label 0 in PS.134) simuliert.

L0⇔L1 („Exchange" z⇔w) wird durch eine Routine (Label 7 in PS.020) ausgeführt.

Durch Hintereinanderschaltung dieser beiden Operationen erhält man bei gelöschtem Flag F0 die beiden folgenden:

„Roll down"	L2	L1	L0
	t	w	z
L0⇔L1	t	z	w
L1⇔L2	z	t	w

„Roll up"	L2	L1	L0
	t	w	z
L1⇔L2	w	t	z
L0⇔L1	w	z	t

Der „Enter"-Befehl (Label e in PS.057) läuft bei gesetztem und bei gelöschtem Flag F0 nach folgendem Muster ab; dabei wird das Flag F0 stets gelöscht.

„Enter"	L2	L1	L0
	(t)	w	z
CF0	w	(t)	z
L1⇔L2	w	z	z
L0⇔L1			

Bei gesetztem Flag F0 (d. h. der Inhalt von L2 wird gleich dem von L1 angenommen) sind zwei besondere Abläufe erforderlich:

„Roll down"
L0 ↔ L1
„Enter"

L2	L1	L0
(t)	w	z
(t)	z	w
z	w	w

„Exchange" z w,
„Roll up"
„Roll down"
L1 L2

L2	L1	L0
(t)	w	z
z	w	w
w	z	w

Tabelle 1: Programm

	000		056		112		168
01	● LBL 4	Das s-fache	● LBL E	Enter	RTN		R→P
02	*		GSB 8		● LBL b	Subtraktion	LN
03	x⇄y		CFO		x⇄y		c
04	LST x		O LBL 0		CHS		e^x
05	*		x⇄y		x⇄y		R←P
06	GTO 8		P⇄S		CHS		RTN
07	● LBL A	y-Anzeige	STO i		F?2	Vorzeich.-	● LBL 2 cos
08	SF2		x⇄y		RTN	wechsel	CF3
09	RTN		P⇄S		● LBL a	Addition	F?2
10	– x –		STO i		RCLi		GTO 2 cosh
11	x⇄y		RTN		+		GTO 0
12	CF2		R/S		x⇄y		● LBL 1 sin
13	O LBL 8	Prefix	● LBL d	Division	P⇄S		SF3
14	x⇄y		STO 2		RCL i		F?2
15	RTN		STO A		P⇄S		GTO 2 cosh
16	● LBL 9	Roll up	x⇄y		+		GSB 0
17	F?0		STO B		x⇄y		GTO 1
18	GTO 7		CHS		O LBL 0		O LBL 0
19	GSB 8		F?2	Kehrwert	F?0	Zeiger-umschaltg.	x⇄y
20	● LBL 7	Exchange	GTO d		RTN	unterdrückt	CHS
21	P⇄S		SF2		SFO		O LBL 2
22	RCL i		x⇄y		O LBL 8	Zeiger-	e^x
23	P⇄S		● LBL c	Multipl.	DSZ	umschaltg.	R←P
24	RCL i		STO 2		ISZ		e
25	F?0		x⇄y		RTN		SF2
26	GSB 8		RCL i		ISZ		F?3
27	CFO		STO*2		RTN		b
28	R↓		x⇄y		● LBL 5	s-te Potenz	SF2
29	R↓		P⇄S		STO E		d
30	GSB 8		STO 2		R↓		a
31	R↓		*		R→P		.
32	R↓		x⇄y		R↑		5
33	F?2		RCL i		y^x		GSB 4
34	GTO 8		STO*2		x⇄y		RTN
35	RTN		*		RCL E		● LBL 3 tan
36	● LBL B	Store	+		*		F?2
37	x⇄I		RCL 2		GTO 5		GTO 3 tanh
38	R↓		P⇄S		● LBL 6	Drehung	x⇄y
39	GSB 0		STO-2		STO E	um 2 πs	CHS
40	GTO B		R↓		R↓		GSB 3
41	● LBL D	Recall	RCL 2		R→P		O LBL 8
42	R↓		F?2		x⇄y		CHS
43	E		F?2		π		x⇄y
44	R↑		GTO 0		↑		RTN
45	● LBL C	CLz&Recall	R↓		+		O LBL 3
46	x⇄I		d		RCL E		2
47	R↑		GTO 0		*		GSB 4
48	P⇄S		O LBL d	Betrag des	+		e^x
49	RCLi		RCL A	Divisors	O LBL 5		R→P
50	P⇄S		x^2		x⇄y		EEX
51	RCL i		RCL B		R←P		–
52	O LBL B		x^2		RTN		E
53	R↑		+		RCL E		2
54	x⇄I		STO÷2		GTO 6		+
55	R↓		÷		● LBL e	Potenz	d
56	GTO 7		RCL 2		GSB 7		RTN

Das Programm ist mit gelöschtem Flag F2 und Winkelmodus RAD aufzuzeichnen. Der Stern * bedeutet die Multiplikationstaste.

Folgt bei gesetztem Flag F0 eine weitere zweistellige Rechenoperation, so bleibt das Flag gesetzt, und die Zeigerumschaltung wird unterdrückt (PS. 132). – In jedem Fall spart die Simulation Zeit.

Rechenbefehle

Die elementaren Rechenoperationen (Labels a, b, c, d, 4) sind mit zahlentheoretischer Genauigkeit programmiert, d. h. rundungsfehlerfrei (bzw. d mit minimalen Rundungsfehler) im Ganzzahlbereich $|z|, |w| \leqq 10^5$. (Übergang zu Polarkoordinaten ergäbe kürzere Routinen, aber längere Rechenzeiten und (größere) Rundungsfehler.)

Bei zweistelligen Rechenoperationen werden w und z in dieser Reihenfolge zu einer Ergebniszahl c = a + bj verknüpft. – Formeln:

$c = w \pm z$: $a = u \pm x$, $b = v \pm y$.
$c = w \cdot z$: $a = ux - vy$, $b = vx + uy$.
$c = s \cdot z$
c = (s reell):
$a = sx$, $b = sy$.
$c = 1/z$: $a = x/(x^2+y^2)$,
$b = -y/(x^2+y^2)$.
$c = W/z$: $a = (ux+vy)/(x^2+y^2)$,
$b = (vx-uy)/(x^2+y^2)$.

Der Tastenbefehl R→P (Wandlung in Polarkoordinaten) liefert gleich zwei wichtige Funktionen einer komplexen Zahl z:

Betrag $r = |z| = \sqrt{x^2+y^2}$ (im X-Register);

Bogen (Hauptwert) Θ = arc z (im Y-Register).

Der Bogen wird im Bogenmaß gemessen; deshalb Winkelmodus RAD einstellen! Die Arcus-Funktion überstreicht einen Vollwinkel ($-\pi < \Theta \leqq \pi$) und ist daher eine „bessere" Funktion als der Arcustangens Θ = arctan (y/x), der nur in der rechten Halbebene

$(-\pi/2 < \Theta < \pi/2)$

mit ihm übereinstimmt.

Der umgekehrte Befehl R←P liefert die Rechtwinkelkoordinaten:

$x = r \cdot \cos\Theta$, $y = r \cdot \sin\Theta$ von $z = r : e^{\Theta j}$.

Grundlegend für alles folgende ist die Darstellung der komplexen Exponentialfunktion,

$z = e^6 = e^{a+bj} = r \cdot e^{\Theta j}$
$= r \cdot (\cos\Theta + j \sin\Theta)$, $r = e^a$, $\Theta = b$

durch die Tastenfolge e R←P. In Umkehrung hiervon erhält man den komplexen Logarithmus,

$c = \log z$, $a = \ln r = \ln|z|$,
$b = \Theta$ = arc z,

mit der Tastenfolge R→P LN.

Die allgemeine Potenz $c = w^z$ ist definiert durch $w^z = e^{z \cdot \log w}$. Für reellen Exponenten s läßt sich die Potenz $c = z^s$ einfacher berechnen mittels:

$a = r^s \cdot \cos \Theta s; \quad b = r^s \cdot \sin \Theta s.$

Für $s = \frac{1}{n}$ erhält man so den Hauptwert der n-ten Wurzel, $c_0 = \sqrt[n]{z}$. Die n verschiedenen Werte dieser Wurzel erhält man aus dem Hauptwert durch fortgesetzte Drehung um den Nullpunkt, jeweils um den Winkel $2\pi/n$, nach der Formel:

$$c_k = {}_k\sqrt[n]{z} = \sqrt[n]{r} \cdot \left(\cos\frac{\Theta + 2k\pi}{n} + j \cdot \sin \frac{\Theta + 2k\pi}{n}\right), \; k = 0, \ldots, n-1.$$

Die n Wurzelwerte bilden die Ecken eines regelmäßigen Vielecks, was man zur Konstruktion auf Millimeterpapier benutzen kann.

Die programmierten Winkelfunktionen sind:

$$\begin{aligned}
\sin z &= (e^{jz} - e^{-jz})/2j; \\
\cos z &= (e^{jz} + e^{-jz})/2; \\
\tan z &= \sin z / \cos z \\
&= \frac{e^{2jz} - 1}{j \cdot (e^{2jz} + 1)}; \\
\sinh z &= (e^{z} - e^{-z})/2; \\
\cosh z &= (e^{z} + e^{-z})/2; \\
\tanh z &= \sinh z / \cosh z \\
&= \frac{e^{2z} - 1}{e^{2z} + 1}.
\end{aligned}$$

Mehrfachausnützung der Labels

Bei Programmen mit vielen zur Wahl stehenden Funktionen wie hier besteht eine Schwierigkeit darin, mit den 20 Labels des HP-67/97 auskommen zu müssen. Im vorliegenden Programm sind 19 Labels (in beliebiger Reihenfolge) manuell oder als Unterprogramme aufrufbar. Solche Labels mögen Haupt-Labels heißen; sie sind im Programm durch einen vorangestellten fetten Punkt (●) gekennzeichnet. Einige dieser Labels treten an anderen Stellen im Programm auch als Hilfs-Labels auf, welche nur innerhalb einer Routine (nicht mit Tastendruck) aufgerufen werden. Sie sind durch einen vorangestellten Kreis (○) gekennzeichnet. Zwischen Hauptlabel und nächstem RTN kann das gleiche Label als Hilfslabel dienen.

Vorsicht! Damit nicht fälschlich ein Hilfs-Label (statt eines Haupt-Labels) aufgerufen werden kann, ist darauf zu achten, daß zwischen dem Haupt-Label Λ und einem ihm nachfolgenden gleichen Hilfs-Label Λ kein RTN-Befehl steht. (Andernfalls würde man nach einem Stopp auf diesem RTN-Befehl mit GSB Λ ein Hilfs-Label anlaufen.) Aus diesem Grunde enden einige unserer Routinen erst nach einem kürzeren oder längeren GTO-Sprung mit RTN.

Eine andere Methode (hier bei sechs Labels angewandt) besteht darin, durch eine „Prefix-Taste (hier A) ein Flag (F2) zu setzen, das normalerweise gelöscht ist. Mit Hilfe des Flags wird in einer anschließend aufgerufenen Routine verzweigt; ihr Label wird also doppelt genützt. Die Speicherbelegung sieht so aus:

Tabelle 2: Bedienung

Nr. Benutzeranweisung	Daten	Tasten	Veränderung der Größen t	w	z	c_m	Rechenzeit[s]
1 Programm laden.							
(2 Aufzeichnen mit:		RAD CF2)					
3 Wahlweise → 4, ..., 34.							
Transporte	b	↑					
4 Komplexe Zahl c = a + bj eingeben.	a		t	w	z		
5 „Enter".		E	w	z	z		1.5
6 „Exchange" z⇄w.		GSB 7	t	z	w		1.5
7 „Roll down".[3]		A GSB 7	z	t	w		*+1.5
8 „Roll up".		GSB 9	w	z	t		2
9 „Store" in C_m (m = 3, ..., 9).	m	B				z	1.5
10 „CLz & Recall" (ohne Stacklift).	m	C	t	w	c_m		1.5
11 „Recall" aus C_m (mit Stacklift).	m	D	w	z	c_m		2.5
12 Pausenanzeige des Imaginärteils.		A R/S					*+6
Rechenoperationen ohne „Prefix"							
13 Addieren.		a	t	t	w+z		1.5
14 Subtrahieren.		b	t	t	w–z		1.5
15 Multiplizieren.		c	t	t	w · z		2
16 Dividieren.		d	t	t	w/z		3
17 Potenzieren.		e	t	t	w		6.5
18 Das s-fache (s reell).	s	GSB 4			z · s		1
19 s-te Potenz (n-te Wurzel für s = 1/n).[4]	s	GSB 5			z		3.5
20 Drehung um 2πs.[5]	s	GSB 6			$r \cdot e^{i(\theta+2\pi s)}$		3
21 Sinus.		GSB 1			sin z		7
22 Cosinus.		GSB 2			cos z		6.5
23 Tangens.		GSB 3			tan z		7
Rechenoperationen mit „Prefix"[3]		A					*=0.5
24 Vorzeichenwechsel.		A b			–z		*+0.5
25 Kehrwert.		A d			1/z		*+1
26 Sinus hyperbolicus.		A GSB 1			sinh z		*+7
27 Cosinus hyperbolicus.		A GSB 2			cosh z		*+6.5
28 Tangens hyperbolicus.		A GSB 3			tanh z		*+7
Ohne Programmunterstützung							
29 Addition einer reellen Zahl s.	s	+			z+s		0.1
30 Das j-fache (j = $\sqrt{-1}$).		x⇄y CHS			z·j		0.1
31 Der j-te Teil.		CHS x⇄y			z/j		0.1
32 Konjugiert komplexe Zahl $\bar{z}$=x–yj.		x⇄yCHSx⇄y			$\bar{z}$		0.1
33 Exponentialfunktion.		e R←P			e^z		1.3
34 Logarithmus.		R→P LN			log z		1.5
35 Neue Rechnung: →3.		CL REG P⇄S					
36 Gesamtlöschung; anschließend →3.		CL REG CF2					

Warnung: Die Berechnung der komplexen Winkelfunktionen dupliziert w, d. h. t nimmt dann den Wert von w an.

[3]) Anstelle des Prefix A kann auch SF2 genommen werden (Zeitersparnis in einem Oberprogramm).

[4]) Im Anschluß an die Berechnung des Hauptwertes von $\sqrt[n]{z}$ liefert wiederholtes R/S (jeweils durch Drehung um $2\pi/n$) der Reihe nach die übrigen Werte der n-ten Wurzel.

[5]) Mit diesem Befehl kann man (mittels Drehung um $2k\pi/n$) den k-ten Wurzelwert c_k auch direkt aus dem Hauptwert c_0 der n-ten Wurzel berechnen: Mit c_0 in LO taste ein: k/n GSB 6.

RA, RB:	z=x+yj nach Division durch z;
RC, RD:	frei;
RE:	2πs für Drehung;
RI:	„Zeiger" i;
R0, RS0: R1, RS1:	Stacklevels L1, L2.
R2, RS2:	Belegt.
R3, RS3: ... R9, RS9:	Komplexe Zahl $c_m = a_m + b_m j$ (m = 3, ..., 9).
Benutzte Flags:	F0, F2, F3.
Winkelmodus:	RAD.

Rechenbeispiele

Zur Prüfung, ob das Programm richtig geladen ist, kann diese Tastenfolge dienen, die jede Routine durchläuft. Sie berechnet den Ausdruck

$$t = (\sin z)^{\tanh w} - \sqrt[5]{z} \text{ für}$$

w = 5-2j und z = 3+4j zu

t = –3.681639844 – 28.39253907 j.

Vorsicht! Tasten dürfen erst gedrückt werden, wenn ein Rechenvorgang beendet ist; sonst treten Fehler auf.

Daten	Tasten
2 CHS ↑ 5	E
4 ↑ 3 ↑ 3	B
	GSB 1 GSB 9
	A GSB 3
	e
3	D
.2	GSB 5 R/S
	b

Um ein eigenes Oberprogramm schreiben zu können, löscht man die nicht benötigten Programmteile, etwa die Winkelfunktionen (ab PS.176). Zu diesem Zweck bereitet man eine Löschkarte vor, die nur einen Befehl (RTN in PS.224) enthält. Dann drückt man

GTO : .175 MERGE

und liest die Löschkarte ein. (Nach etwaiger Error-Anzeige beim Einlesen muß stets erneut MERGE gedrückt werden!)

Der Benutzer hat nun eine Programmebene, drei Labels (1, 2, 3) und ein Flag (F1) zu freier Verfügung. Benötigt man mitten in einer Rechnung wiederum die Winkelfunktion, dann liest man die Programmkarte (Tab. 1) mit RTN MERGE erneut ein. Auf diese Weise bleibt der Programmstatus (Flagzustände, usw.) erhalten.

Als Beispiel für ein Oberprogramm hier die Berechnung des komplexen Polynoms

$w = z^3 + c_3 z^2 + c_4 z + c_5$

nach dem Horner-Verfahren:

$w = ((z + c_3) \cdot z + c_4) \cdot z + c_5$.

Die Größen c_3, c_4, c_5, z seien in C3, C4, C5, L0 gespeichert. Das Oberprogramm lautet:

e 3 D a c 4 D a c 5 D a RTN.

Dieses Programm ohne Labels kann man auch am Anfang des Programmspeichers laden (im RUN-Zustand RTN drücken) und mit RTN R/S starten. Für $c_3 = 3 + 4j$, $c_4 = 5 + 6j$, $c_5 = 7 + 8j$ und z = 1 + 2j ergibt sich: w = –36 + 22j (Rechenzeit: 14 s).

Literatur

[1] HP-67 Complex Stack. PPC Journal, V6N1P33 (1979). Herausgeber: Richard Nelson, 2541 W, Camden Place, Santa Ana, CA 92704, USA.

Binär-Dezimal-Umwandlung mit dem 6502

Will man z. B. den Inhalt der Timer des AIM-65 dezimal anzeigen oder weiterverarbeiten, so ist eine Binär-Dezimal-Umwandlung erforderlich, da man die Timer nicht veranlassen kann, dezimal zu zählen.

Das im *Bild* gezeigte Programm wandelt eine maximal achtstellige Hex-Zahl in die entsprechende maximal zehnstellige Dezimalzahl (BCD-Code) um. Die Stellenzahl läßt sich jedoch leicht beliebig erweitern oder reduzieren.

Das Programm arbeitet nach einem Schema, das in [1] ausführlich beschrieben ist. Dabei wird die Binärzahl, beginnend mit dem höchstwertigen Bit, von rechts nach links in einen „BCD-Rahmen" geschoben. Überschreitet eine Stelle den Wert 4 (0100), so muß vor dem nächsten Schiebevorgang eine Drei an dieser Stelle addiert werden. Schiebevorgang und, falls nötig, Korrektur wechseln sich ab, bis das letzte Bit der Binärzahl in den BCD-Rahmen geschoben ist. Das Ergebnis steht dann in der richtigen Reihenfolge in den Speicherzellen 0000 bis 0004 (Ergebnisspeicher), d. h. im niederwertigen Halb-Byte von 0004 stehen die Einer, im höherwertigen Halb-Byte die Zehner usw. Die umzuwandelnde Binär- bzw. Hex-Zahl muß in den Speicherzellen 0005 bis 0008 stehen (Hex-Speicher), wobei ebenfalls das niederwertigste Byte in 0008 steht.

Will man das Programm für andere Stellenzahlen abändern, so ist folgendes zu beachten: In Zeile 0391 wird das Indexregister X mit der Stellenzahl der Binärzahl geladen (die Null wird mitgezählt). Da die Stellenzahl der Hex-Zahlen meist kleiner ist als die der Dezimalzahlen, müssen für diese genügend Speicherplätze reserviert werden (die größtmögliche Dezimalzahl ist $16^n - 1$, wobei n die maximale Stellenzahl der Hex-Zahl bedeutet). Das Indexregister Y dient als Zeiger auf das jeweils zu prüfende Byte des Ergebnisspeichers. Es wird in Zeile 0393 mit der Zahl der erforderlichen Speicherplätze für den Ergebnisspeicher minus eins geladen.

Für jedes zusätzliche Byte im Hex- und Ergebnisspeicher muß ein weiterer ROL-Befehl eingefügt werden, bzw. kann bei Reduzierung der Stellenzahl weggelassen werden.

Für den Zugriff auf den Hex-Speicher und den Ergebnisspeicher wurde die direkte Adressierung gewählt, damit diese Speicher leicht in einen anderen Speicherbereich verlegt werden können.

Udo Wendel

Literatur

[1] *Tietze/Schenk:* Halbleiter-Schaltungstechnik, 5. Auflage, Springer-Verlag, Seite 459 ff.

Unterprogramm zur Umwandlung von Hex-Zahlen in Dezimalzahlen mit dem 6502

```
0380 A9 LDA #00
0382 8D STA 0000
0385 8D STA 0001
0388 8D STA 0002
038B 8D STA 0003
038E 8D STA 0004
0391 A2 LDX #1F
0393 A0 LDY #04
0395 0E ASL 0008
0398 2E ROL 0007
039B 2E ROL 0006
039E 2E ROL 0005
03A1 2E ROL 0004
03A4 2E ROL 0003
03A7 2E ROL 0002
03AA 2E ROL 0001
03AD 2E ROL 0000
03B0 E0 CPX #00
03B2 F0 BEQ 03DB
03B4 B9 LDA 0000,Y
03B7 48 PHA
03B8 29 AND #0F
03BA C9 CMP #05
03BC 90 BCC 03C0
03BE 69 ADC #02
03C0 99 STA 0000,Y
03C3 68 PLA
03C4 29 AND #F0
03C6 C9 CMP #50
03C8 90 BCC 03CC
03CA 69 ADC #2F
03CC 19 ORA 0000,Y
03CF 99 STA 0000,Y
03D2 88 DEY
03D3 C0 CPY #FF
03D5 D0 BNE 03B4
03D7 CA DEX
03D8 4C JMP 0393
03DB 60 RTS
```

Richard Koellner

Netzteil-berechnung in Basic

Die Bemessung von Netzteilen gehört zu den Bereichen, die von Hobby-Elektronikern etwas vernachlässigt werden. Meistens wird auf fertig gewikkelte Transformatoren zurückgegriffen. Andererseits hat fast jeder Bastler irgendwo ausgebaute Transformatoren herumliegen, die sich sehr gut zum Umwickeln eignen; man müßte nur wissen, wie! Dabei hilft das folgende Programm, das, ausgehend von den gewünschten Sekundärdaten, einen Transformator sowie den Gleichrichter und den Ladekondensator vollständig berechnet.

Damit das Programm nicht zu lang und zu unübersichtlich wird, beschränkt es sich auf die häufigsten Anwendungsfälle: einzelne und duale Spannungen mit Brückengleichrichtung und Ladekondensator (*Bild 1*). Eine nachfolgende elektronische Stabilisierung bleibt davon unberührt. Von diesen zwei Wicklungstypen dürfen hier bis zu zehn, beliebig gemischt, bei einem Transformator auftreten. Für jedes Sekundärteil müssen die Ausgangsspannung bei maximal zulässiger Belastung U_a (in V), der maximale Ausgangsstrom I_a (in A) und die maximale Brummspannung U_{Br} (in V_{ss}) angegeben werden; bei dualen Spannungen nur die Werte eines Zweiges. Es werden dann zunächst die zu übertragende Gesamtleistung berechnet und die möglichen Kerne zur Auswahl gestellt. (Oft steht der kleinstmögliche Kern nicht zur Verfügung, so daß ein stärkerer Kern verwendet werden muß.) Die Abmessungen der Kerne sind in *Bild 2* angegeben. Die zu den Kerntypen gehörenden Parameter sind im DATA-Block gespeichert, ihre Bedeutung ist (v.l.n.r.): Kerntyp, maximal übertragbare Leistung, Verlustfaktor, primäre Windungszahl bei 220 V, primärer Drahtdurchmesser in mm, sekundäre Windungszahl in Wdg./V, sekundärer Drahtdurchmesser in mm/$\sqrt{A}$. Die Parameterliste kann nach eigenen Vorstellungen erweitert werden.

Damit sind alle notwendigen Angaben gemacht und die Ausgabedaten können berechnet werden (vgl. Tietze/Schenk, „Grundlagen der Halbleiter-Schaltungstechnik", Springer-Verlag). Für jede Sekundärwicklung erhält man die Nennspannung und die Windungszahl sowie den Nennstrom und den Drahtdurchmesser. Weiter wird noch die Leerlaufspannung des Netzteils angegeben (das ist zugleich die erforderliche Spannungsfestigkeit des Ladekondensators), die Ladekapazität sowie der (periodische) Spitzenstrom im Gleichrichter. Damit ist das Netzteil vollständig beschrieben. Das Programm ist in *Bild 3* aufgelistet, ein Beispiel für eine Anwendung gibt *Bild 4*. Die berechneten Werte sind Idealwerte, d. h. man kann etwas abweichende Werte nehmen, ohne größere Probleme zu befürchten. (Ist zum Beispiel als Drahtdurchmesser 1,03 mm angegeben, dann ist 1 mm der angemessene Wert.) Für C_L wird man im allgemeinen den nächstgrößeren Normwert verwenden.

Drei kurze Bemerkungen noch zum Programm:

1. Es wird die nicht in allen Basic-Systemen vorhandene formatierte Ausgabe benützt (PRINT USING), wobei # ein Maskenelement für eine Ziffer ist

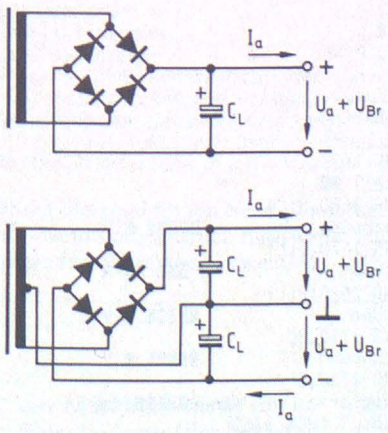
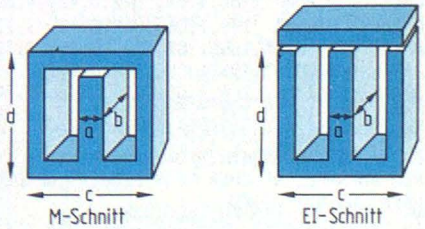


Bild 1. Netzteil-Standardschaltungen für duale und einfache Ausgangsspannungen



Typ	a	b	c	d
M42	12	15	42	42
M55	17	21	55	55
M65	20	27	65	65
M74	23	32	74	74
M85a	29	32	85	85
M85b	29	45	85	85
M102a	34	35	102	102
M102b	34	52	102	102
EI130a	35	52	130	87,5
EI130b	35	36	130	87,5
EI150a	40	40	150	100
EI150b	40	50	150	100

Bild 2. Abmessungen unterschiedlicher Trafo-Kerntypen

```
  10  REM BERECHNUNG VON NETZTEILEN
  20  REM
  30  REM INITIALISIERUNG
  40  DIM P(10),U(10),I(10),B(10),V(10),G(10)
  50  PO=0
  60  REM
  70  REM
 100  REM EINGABE DER NETZTEIL- (SEKUNDAER-) DATEN
 110  REM BEI DUALEN SPANNUNGEN NUR EIN ZWEIG
 120  PRINT "Wieviele Sekundaerwicklungen?"
 130  INPUT Z
 140  IF Z .GT. 10 THEN 160
 150  IF Z .GT.  0 THEN 180
 160  PRINT "Bitte nur Wert zwischen 1 und 10 !"
 170  GOTO 120
 180  FOR L=1 TO Z
 190  PRINT USING 200,L
 200 :Sekundaerteil ##, Kenndaten-Eingabe
 210  PRINT "Art des Sekundaerteils (bitte Kennzahl eingeben)"
 220  PRINT "      (1) einzelne Ausgangsspannung"
 230  PRINT "      (2) duale Ausgangsspannungen"
 240  INPUT G(L)
 250  IF G(L) .EQ. 1 THEN 280
 260  IF G(L) .NE. 2 THEN 210
 270  PRINT "Bitte im folgenden nur die Daten  e i n e r
      Teilspannung angeben"
 280  PRINT
 290  PRINT "Ausgangsspannung bei maximaler Belastung (V)?"
 300  INPUT U(L)
 310  U(L)=ABS(U(L))
 320  PRINT "Maximaler Ausgangsstrom (A)?"
 330  INPUT I(L)
 340  I(L)=ABS(I(L))
 350  PRINT "Brummspannung? (maximale Belastung, Vss)"
 360  INPUT B(L)
 370  B(L)=ABS(B(L))
 380  IF B(L) .LT. U(L)/2 THEN 410
 390  PRINT " +++ Wert zu gross!"
 400  GOTO 350
 410  IF G(L) .EQ. 1 THEN 470
 420  U(L)=2*U(L)
 430  B(L)=2*B(L)
 440  REM
 450  REM ERSTE BERECHNUNGEN
 460  REM AUSGANGSSPANNUNG BEI IDEALER GLAETTUNG
 470  V(L)= U(L) + 0.666*B(L)
 480  REM NENNLEISTUNG
 490  P(L)=1.25*I(L)*(V(L)+1.4)
 500  REM GESAMTLEISTUNG DES TRAFOS
 510  PO=PO+P(L)
 520  NEXT L
 530  REM
 540  REM
 600  REM KERNTYP DES TRAFOS
 610  REM PARAMETERLISTE
 620  DATA  " M 42"   ,  4,1.31,4716,0.09,28.00,0.61
 630  DATA  " M 55"   , 15,1.20,2671,0.18,14.62,0.62
 640  DATA  " M 65"   , 33,1.14,1677,0.26, 8.68,0.64
 650  DATA  " M 74"   , 55,1.11,1235,0.34, 6.24,0.65
 660  DATA  " M 85a"  , 80,1.09, 978,0.42, 4.83,0.66
 670  DATA  " M 85b"  ,105,1.06, 655,0.48, 3.17,0.67
 680  DATA  " M 102a",135,1.07, 763,0.56, 3.72,0.69
 690  DATA  " M 102b",195,1.05, 513,0.69, 2.45,0.71
 691  DATA  "EI 130a",264,1.05, 726,0.92, 3.50,0.80
 692  DATA  "EI 130b",335,1.05, 572,1.06, 2.70,0.82
 693  DATA  "EI 150a",387,1.04, 572,1.17, 2.70,0.85
 694  DATA  "EI 150b",473,1.04, 440,1.30, 2.15,0.85
 700  REM ABFRAGE, WELCHER KERN BENUTZT WERDEN SOLL
 710  PRINT USING 720,PO
 720 :Zu uebertragende Gesamtleistung des Trafos:##### W
 730  IF PO .LE. 473 THEN 760
 740  PRINT " +++ Fuer diese Leistung ist kein Kern bekannt!"
 750  STOP
 760  PRINT "Welcher der folgenden Kerne soll verwendet werden?"
 770  PRINT "          (bitte Schluesselzahl eingeben)"
 780  RESTORE
 790  K1=0
 800  FOR L=1 TO 12
 810  READ K&,P2,F,W1,D1,W2,D2
 820  IF P2 .LT. PO THEN 860
 830  PRINT USING 840,L,K&,P2
 840 :       (##)   Typ: ".....", maximale Leistung:#### W
 850  GOTO 870
 860  K1=L
 870  NEXT L
 880  K1=K1+1
 890  INPUT K2
 900  IF K2 .LT. K1 THEN  920
 910  IF K2 .LT. 13 THEN 1015
 920  PRINT " +++ Ungueltiger Eingabewert!"
 930  GOTO 760
 940  REM EINGABEN BEENDET
 950  REM
 960  REM
1000  REM BERECHNUNG UND AUSGABE
1010  REM PRIMAERWICKLUNG
1015  RESTORE
1020  FOR L=1 TO K2
1030  READ K&,P2,F,W1,D1,W2,D2
1040  NEXT L
1050  PRINT
1060  PRINT "PRIMAERWICKLUNG"
1070  PRINT "----------------"
1075  PRINT "Kerntyp: ",K&
1080  PRINT USING 1090,W1,D1
1090 :Windungszahl bei 220 V:######, Drahtdurchmesser:##.## mm
1100  REM
1110  REM SEKUNDAERWICKLUNGEN
1120  PRINT
1130  PRINT "SEKUNDAERWICKLUNGEN"
1140  PRINT "--------------------"
1150  FOR L=1 TO Z
1160  REM ITERATION ZUR BESTIMMUNG DER NENNSPANNUNG
1170  U2=U(L)
1180  L1=0
1190  R2= U2*U2 / P(L)*(F-1)
1200  Q=SQR(R2*I(L)/(2*V(L)))
1210  S=V(L)-(1.414*U2*F - 1.4)*(1-Q)
1220  IF ABS(S) .LT. V(L)/1000 THEN 1320
1230  L1=L1+1
1240  IF L1 .GT. 20 THEN 1280
1250  U2=U2+S
1260  GOTO 1190
1270  REM FEHLERHAFTE ITERATION
1280  PRINT " +++ Die Werte der naechsten Sekundaerwicklung
      sind wegen"
1290  PRINT "     fehlerhafter Iteration moeglicherweise
      ungenau!"
1300  REM
1310  REM NENNSTROM
1320  I2=P(L)/U2
1330  REM WICKLUNGSZAHL
1340  W=W2*U2
1350  REM DRAHTDURCHMESSER
1360  D=D2*SQR(I2)
1370  REM LADEKONDENSATOR (MIKROFARAD)
1380  C=I(L)/(100*B(L))*(1-SQR(Q))*1E6
1390  REM LEERLAUFSPANNUNG
1400  UO=V(L)/(1-Q)
1410  REM SPITZENSTROM IM GLEICHRICHTER
1420  I3=UO*Q/R2
1430  REM
1440  REM DUALE SPANNUNGEN
1450  IF G(L) .EQ. 1 THEN 1610
1460  U(L)=U(L)/2
1470  UO=UO/2
1480  B(L)=B(L)/2
1490  C=C*2
1500  REM
1510  REM
1600  REM ENDAUSGABE
1610  PRINT
1620  PRINT USING 1630,L
1630 :Sekundaerteil ##
1640  PRINT USING 1650,U(L)
1650 :Ausgangsspannung          #####.# V
1660  PRINT USING 1670,I(L)
1670 :Ausgangsstrom              ###.## A
1680  PRINT USING 1690,B(L)
1690 :Brummspannung             #####.# Vss
1700  PRINT USING 1710,UO
1710 :Leerlaufspannung          #####.# V
1720  PRINT USING 1730,C
1730 :Ladekondensator        mind#######.# uF
1740  IF G(L) .EQ. 1 THEN 1760
1750  PRINT "(Die obigen Werte beziehen sich auf eine der
      Teilspannungen)"
1760  PRINT USING 1770,U2
1770 :Erforderliche Nennspannung der (gesamten) Wicklung:
      #####.# V
1780  PRINT USING 1790,W
```

```
1790 :     daraus folgt die Sekundaer-Windungszahl:    ■■■■■■■.■ Wdg
1800  IF G(L) .EQ. 1 THEN 1840
1810  W=W/2
1820  PRINT USING 1830,W
1830 :         (Mittelanzapfung bei■■■■■■■.■ Wdg)
1840  PRINT USING 1850,I2
1850 :Nennstrom der Sekundaerwicklung:                ■■■.■■■ A
1860  PRINT USING 1870,D
1870 :     daraus folgt der Drahtdurchmesser:        mind■■■.■■ mm
1880  PRINT USING 1890,I3
1890 :Maximale Belastung des Gleichrichters:          ■■■.■■■ A
1900  NEXT L
2000  END
```

Bild 3. Listing des Basic-Programms. Die bei der vorhandenen Basic-Implementierung vorhandenen Sonderzeichen sind dem Text zu entnehmen

```
PRIMAERWICKLUNG

Kerntyp:  M 74
Windungszahl bei 220 V: 1235, Drahtdurchmesser:  .34 mm

SEKUNDAERWICKLUNGEN


Sekundaerteil  1
Ausgangsspannung                  15.0 V
Ausgangsstrom                      .50 A
Brummspannung                     2.0 Vss
Leerlaufspannung                 19.5 V
Ladekondensator          mind    1485 uF
Erforderliche Nennspannung der (gesamten) Wicklung:    13.4 V
     daraus folgt die Sekundaer-Windungszahl:          83.3 Wdg
Nennstrom der Sekundaerwicklung:                        .830 A
     daraus folgt der Drahtdurchmesser:          mind   .59 mm
Maximale Belastung des Gleichrichters:                 1.818 A

Sekundaerteil  2
Ausgangsspannung                   8.0 V
Ausgangsstrom                      .50 A
Brummspannung                     2.0 Vss
Leerlaufspannung                 11.2 V
Ladekondensator          mind    1473 uF
Erforderliche Nennspannung der (gesamten) Wicklung:     8.0 V
     daraus folgt die Sekundaer-Windungszahl:          50.2 Wdg
Nennstrom der Sekundaerwicklung:                        .834 A
     daraus folgt der Drahtdurchmesser:          mind   .59 mm
Maximale Belastung des Gleichrichters:                 1.783 A

Sekundaerteil  3
Ausgangsspannung                   8.0 V
Ausgangsstrom                     1.50 A
Brummspannung                     2.0 Vss
Leerlaufspannung                 11.2 V
Ladekondensator          mind    4419 uF
Erforderliche Nennspannung der (gesamten) Wicklung:     8.0 V
     daraus folgt die Sekundaer-Windungszahl:          50.2 Wdg
Nennstrom der Sekundaerwicklung:                       2.501 A
     daraus folgt der Drahtdurchmesser:          mind  1.03 mm
Maximale Belastung des Gleichrichters:                 5.349 A
```

Bild 4. Beispiel für ein Netzteil mit drei Sekundärteilen. Die errechneten Werte sind natürlich Idealwerte, und man wird jeweils den nächstliegenden Normwert verwenden

und „.....“ ein Maskenelement für einen String. Dies dient nur der besseren Übersichtlichkeit und kann selbstverständlich auch anders gelöst werden.

2. Bei der Iteration zur Berechnung der Nennspannung wird zur Vermeidung von endlosen Schleifen nach 20 Schritten abgebrochen. Dieser Fall trat allerdings beim Verfasser bisher noch nicht auf.

3. Bei den meisten üblichen Tischcomputern sind folgende Implementierungen auszutauschen: 1 statt .GT., < statt .LT., > = statt .GE., < statt .LE., # statt ⊞

1200 Bd per Funk

Nachdem die Deutsche Bundespost den Funkamateuren die Übertragung beliebiger Fernschreib-Codes mit der Auflage zugestanden hat, ihre Rufzeichen entweder in Sprache oder Morsetelegrafie zu nennen, ist auch das Übertragen z. B. von Mikrocomputer-Programmen per Funk möglich.

Beschränkt man Betrachtungen hierüber auf den UKW-Bereich, da nur hier eine ausreichende Störsicherheit gewährleistet werden kann, so bleiben als mögliche Modulationsarten im wesentlichen nur Einseitenband-Modulation (SSB) und Frequenzmodulation übrig.

Während SSB eine Nf-Bandbreite von etwa 2,5 kHz bietet, lassen sich mit FM-Geräten auch noch rund 5 kHz (wenn auch mit einem gewissen Amplitudenabfall) übertragen. Versuche des Verfassers zeigten, daß sich bei SSB eine Übertragung mit 300 Bd im Kansas-City-Format problemlos durchführen läßt.

Bei FM kann sogar eine Übertragungsgeschwindigkeit von 1200 Bd verwendet werden; hierzu diente versuchsweise das von AIM-65 und PC-100 verwendete Kassettenformat. Hierbei ist es nämlich möglich, die Kassetten-Ein- und Ausgänge des Computers mit den Nf-Anschlüssen des Funkgerätes zu verbinden. Besonders gute Ergebnisse wurden erzielt, als der AIM-Kassetteneingang nicht direkt am Lautsprecherausgang des Funkgeräts, sondern am Kollektor des Eingangstransistors im PLL-Konverter aus Heft 16/1979 angeschlossen war, da seine Begrenzereigenschaften für einen ausgeglichenen Pegel sorgen. Wie bei der Aufzeichnung von AIM-Programmen auf Kassette ist der „Gap“-Wert in Zelle A409 bei der Übertragung von Basic-Programmen auf mindestens etwa 20 zu erhöhen. Für Maschinenprogramme genügt dagegen ein Wert von 02...08.

Zugegebenermaßen ist die Übertragung von 1200 Bd nicht gerade unkritisch, und es werden hohe Anforderungen an Frequenz- und Phasengang von Sender und Empfänger gestellt. In den meisten Fällen wird es deshalb auch erforderlich sein, Korrekturen des Nf-Frequenzganges durch geeignete RC-Glieder empirisch vorzunehmen. Beim Kansas-City-Format (300 Bd) ist das nicht so kritisch; geeignete Software hierfür findet sich in unserem Sonderheft „Mikrocomputer-Anwendungen“ für AIM-65 und PC-100. Fe.

Basic-Editieren leicht gemacht

Basic-Programme im AIM-65/PC-100-Texteditor

Der Basic-Interpreter von AIM-65 und PC-100 verfügt leider nicht über die komfortablen Änderungs- und Editiermöglichkeiten des ROM-Texteditors, der Bestandteil des Monitorprogrammes ist. Es ist aber ohne weiteres möglich, Basic-Programme im Texteditor aufzubereiten. Dabei sind folgende Dinge zu beachten:

1. Die im Texteditor stehende erste Basic-Zeile darf nur einen Leerraum enthalten.
2. Die letzte Zeile darf nur ein CTRL-Z (hex 1A) enthalten. Dieses Zeichen ist im Display unsichtbar, erscheint aber auf dem Drucker als Z.
3. Im Texteditor stehende Direktbefehle, wie NEW, RUN, POKE usw., d. h. Befehle ohne vorangestellte Zeilennummer, werden beim Laden des Programms sofort ausgeführt. So ist es z. B. möglich, mit RUN das Programm automatisch zu starten.
4. Es ist möglich, von Basic aus nur einen Teil des Programms zu laden. Dazu schreibt man im Texteditor nur vor den gewünschten Programmteil eine Zeile, die nur aus einem Leerraum besteht, und dahinter ein CTRL Z.
5. Wenn der Editor zwei Zeilen mit gleicher Zeilennummer enthält, wird nur die zweite von Basic gespeichert. Die erste braucht im Editor deshalb nicht unbedingt gleich gelöscht zu werden, wenn man „nur mal eben" eine andere Version dieser Zeile ausprobieren will.
6. Kommentare (REMs) ohne vorangestellte Zeilennummer im Editor erleichtern den Überblick, werden beim Laden aber von Basic nicht mitgespeichert – eine sehr effektive Weise, Kommentare zu schreiben, die keinerlei Speicherplatz kosten!

Selbstverständlich ist es auch möglich, wie beim Assembler den Umweg über die Kassette zu vermeiden und einen Editor-Text direkt aus dem Speicher in den Basic-Interpreter zu übernehmen. Dies geschieht folgendermaßen:

1. Der User-Input-Vektor muß wie folgt gesetzt werden: 0108=D0; 0109=FA.
2. Beim Initialisieren von Basic ist für Memory Size der Wert 2047 einzugeben. Später sollte man nur noch mit Taste 6 in Basic zurückgehen.
3. Den Editor kann man von 800 bis 1000 initialisieren und nun das Programm in den Texteditor eingeben oder laden. (Nicht vergessen: Erste Zeile = Leerraum, letzte Zeile = CTRL Z!) Den Editor darf man nur mit der Tastenfolge T und Q wieder verlassen.
4. Nun kann man über Taste 6 in Basic zurückgehen und LOAD eingeben. Auf IN= hat man mit U zu antworten. Das Programm wird jetzt in den Basic-Interpreter übernommen.

Die geschilderte Methode ist ideal geeignet, um Programme bis zu einer Länge von etwa 2 KByte zu entwickeln, da alle Editor-Befehle, speziell die Find- und Change-Funktionen, zur Verfügung stehen. Deshalb muß man nicht jede zu ändernde Zeile komplett neu eingeben.

Diese Vorgehensweise ähnelt sehr den Umgang mit komfortablen Compilern, bei denen die Programme ebenfalls zunächst mit dem Texteditor erstellt und dann erst „übersetzt" werden.

Noch ein kleiner Tip für Leute, die nach dem Druck auf Taste 6 plötzlich die Frage Memory Size = ? auf dem Display wiederfinden und befürchten, das Basic-Programm im Speicher sei nun gelöscht. Nein, keineswegs! Drücken Sie aber um Himmels willen jetzt nicht Return, sondern Escape und ändern Sie den Inhalt zweier Zellen: 0001=00, 0002=B9. Mit LIST kann man sich das verlorengeglaubte Basic-Programm ansehen, nachdem man nochmals Taste 6 gedrückt hat. (Mit freundlicher Genehmigung der Fa. Rockwell übersetzt aus „Interactive" Nr. 3.) Fe.

PET-Menue

Da der PET leider nicht über ein eingebautes Zählwerk an der Kassette 1 verfügt, bleiben nur zwei Möglichkeiten, schnellen Zugriff zu einem bestimmten Programm auf einer C-60 zu bekommen: indem ein Bandzählwerk eingebaut wird, oder indem der Zugriff per Software erfolgt.

Das im *Bild* aufgelistete Programm arbeitet sehr einfach: Nach einer Abfrage, ob am Recorder eine Taste gedrückt ist, wird zunächst der Recorder in die definierte Anfangsstellung gebracht, und es erscheint ein Inhaltsverzeichnis über die auf der Kassette aufgezeichneten Programme 0...9.

Durch einen einfachen Tastendruck wird nun das Programm gewählt und der Recorder spult im schnellen Vorlauf an den Anfang des gesuchten Programms. Es muß nur noch von FFWD auf PLAY geschaltet werden, das Programm wird geladen und startet automatisch.

Das Programm ermittelt selbständig, ob es auf einem PET oder CBM läuft, und paßt die Zeitdauer des schnellen Vorlaufs entsprechend der RAM-Bestückung an. Michael Penzkofer

```
2 PT=PEEK(50003):IFPT=0THEN5
3 P1=249:Q1=158:Q2=159:Q3=623:P9=53:GOTO9
5 P1=519:Q1=525:Q2=526:Q3=527:P9=135
9 POKE59468,14
10 PRINT"sQQQQQ"TAB(5)"Druecke STOP-Taste am Recorder #1
20 X$=""
150 POKEP1,52
155 IFPEEK(P1)<>0THEN150
160 PRINT"sQ"TAB(5)"Druecke RF.FWD)>r auf Kasette 1
200 PRINT"Q Waehle die Programm-Nr...":PRINT
210 FORJ=0TO9:READB$,A$:PRINT:PRINTTAB(8);B$;" -";"     ";A$:NEXT:WAIT59411,8,8
220 GOSUB800
315 POKE59411,53:T=TI+(X*22.25*PEEK(P9)-12*X*X)
320 IFTI<TTHEN320
330 POKE59411,61-PT
335 PRINT"s
340 PRINT"SQQQQQQQQQQQ      RDruecke STOP
345 WAIT59408,16:POKE59411,61
355 POKE59468,12:PRINT"sQQQQ":POKEQ1,2:POKEQ2,131:POKEQ3,131:POKEQ3+1,131
375 END
390 DATA0,.....
400 DATA1,.....
410 DATA2,.....
420 DATA3,.....
430 DATA4,.....
440 DATA5,.....
450 DATA6,.....
460 DATA7,.....
470 DATA8,.....
480 DATA9,.....
800 GETX$:IFX$<>""ANDX$<>CHR$(13)THEN820
810 POKEP1,52:POKE59411,61-PT:GOTO800
820 X=VAL(LEFT$(X$,1))
830 RESTORE:RETURN
```

Mit dem PET-Menue-Programm läßt sich gezielt auf bestimmte Programme einer Kassette mit schnellem Vorlauf zugreifen

Roland Hölzl

Leitungsberechnung mit dem TI-59

Mit dem hier vorgestellten Programm können von einer Leitung mit einem komplexen Abschlußwiderstand die Dämpfung und die Phase in Abhängigkeit von der Frequenz und der Länge der Leitung berechnet werden. Durch ein Zusatzprogramm kann man darüber hinaus auch noch den Eingangswiderstand dieser Anordnung berechnen.

Um das Übertragungsverhalten einer Leitung berechnen zu können, müssen der Wellenwiderstand und das Übertragungsmaß der Leitung bekannt sein. Der Wellenwiderstand einer Leitung wird folgendermaßen berechnet:

$$\underline{Z}_W = \sqrt{\underline{Z}_K \cdot \underline{Z}_L} = \sqrt{\frac{R' + pL'}{G' + pC'}}$$

wobei $\underline{Z}_K$ der Kurzschlußwiderstand ist, $\underline{Z}_L$ der Leerlaufwiderstand, R' der Widerstandsbelag (Widerstand pro Meter), G' der Ableitungsbelag (Ableitung pro Meter), L' der Induktivitätsbelag (Induktivität pro Meter) und C' der Kapazitätsbelag (Kapazität pro Meter). Das „p" ist hier und in den folgenden Formeln gleich $j2\pi pf$, wobei j ausdrückt, daß es sich um den Imaginärteil einer komplexen Zahl handelt.

Das Übertragungsmaß einer Leitung ist definiert:

$$\underline{\gamma} = \sqrt{(R' + pL') \cdot (G' + pC')}$$

Bei kurzen Leitungen kann man R' und G' Null setzen, dadurch wird der Wellenwiderstand reell (ohne Blindwiderstand) und frequenzunabhängig. Will man nicht gerade Überlandleitungen berechnen und liegt z. B. eine Kupferleitung mit Kunststoffisolierung vor, ist diese Vereinfachung zulässig, und es lassen sich die Leitungsparameter auf einfache Weise festlegen. Man nimmt ein 1 m langes Kabel, schließt ein Ende kurz und mißt die Induktivität (Induktivitätsbelag); dann unterbricht man den Kurzschluß und mißt die Kapazität (Kapazitätsbelag). Kennt man den Wellenwiderstand des Kabels, braucht man nur einen Wert zu messen, denn es gilt:

$$Z_W = \sqrt{\frac{L'}{C'}}$$

Das Übertragungsverhalten einer Leitung wird nach der Formel

$$\frac{\underline{U}_1}{\underline{U}_2} = \cosh \underline{\gamma} l + \frac{\underline{Z}_W}{\underline{Z}_2} \sinh \underline{\gamma} l$$

berechnet, wobei l die Länge der Leitung (in Meter) ist und $\underline{Z}_2$ der Abschlußwiderstand der Leitung. Dieser Abschlußwiderstand kann als Bruch, bestehend aus Zähler- und Nennerpolynom, dargestellt werden.

$$\underline{Z}_2 = \frac{A_0 + A_1 p + \ldots A_n p^n}{B_0 + B_1 p + \ldots B_m p^m}$$

Programm- und Dateneingabe

Mit dem Programm (*Bild 1*) für den Rechner TI-59 mit Standard-Software-Modul und dem Drucker PC-100 lassen sich beliebige Leitungen mit einem komplexen Abschlußwiderstand berechnen, wobei die Länge der Leitung und die Frequenz Parameter sind. Für den Abschlußwiderstand $\underline{Z}_2$ gelten folgende Einschränkungen: Der Grad des Zählerpolynoms n bzw. der Grad des Nennerpolynoms m muß kleiner/gleich fünf sein; soll eine Leitung im Leerlauf berechnet werden, z. B. eine sog. ¼-Leitung, darf der Abschlußwiderstand nicht größer als 10^{20} in den Rechner eingegeben werden.

Bevor das Programm in den Rechner eingelesen wird – per Tasten oder Magnetkarten (drei Kartenseiten) –, muß die Speicher-Bereichsverteilung des Rechners auf 639.39 gestellt werden. Da das Programm für den TI-59 eigentlich eine Nummer zu groß ist, mußte ein Teil der Speicher doppelt genutzt werden, das bedeutet, daß sich während des Programmablaufs die Speicher-Bereichsverteilung auf 559.49 verändert.

Alle Daten, die eingegeben bzw. vom Rechner ausgegeben werden, werden mit den Grundeinheiten verwendet bzw. interpretiert – also Ohm, Farad, Henry, Meter und Hertz. Nach Berechnung von $\underline{Z}_2$ wird über Label A der Grad des Zählerpolynoms n eingegeben. Die Koeffizienten des Zählerpolynoms werden der Reihe nach beginnend mit A_0 bis A_n durch wiederholtes Drücken der R/S-Taste dem Rechner eingegeben. Das Nennerpolynom wird ebenso eingegeben, nur wird jetzt der Grad über Label B eingelesen. R' wird über Label C und L', G' und C' werden in dieser Reihenfolge durch wiederholtes Drücken der R/S-Taste eingegeben. Diese Grunddaten werden zur Kontrolle sofort ausgedruckt. Nachdem C' eingelesen wurde, hat sich die Speicher-Bereichsverteilung des Rechners verändert. Sollen jetzt die Grunddaten geändert werden, muß wieder von vorne mit dem Einlesen der Magnetkarten begonnen werden.

Nun folgt die Eingabe der Parameter. Über Label D wird die Startfrequenz eingegeben und über die R/S-Taste die Endfrequenz. Die Anfangslänge wird über Label E eingegeben, die Endlänge wieder über die R/S-Taste. Die Schrittweite, mit der die Frequenz verändert werden soll, wird über Label A', die Schrittweite der Länge wird über die R/S-Taste eingegeben. Will man ein logarithmisches Frequenzraster, wird noch über Label B' die Anzahl der zu berechnenden Werte pro Dekade eingegeben.

Um das Übertragen der Werte auf halblogarithmisches Papier zu erleichtern, rundet das Programm die Frequenzwerte. Deshalb sollte die Anzahl der zu berechnenden Werte pro Dekade nicht größer als 20 gewählt werden, da sonst der Rechner mehrmals gleiche Frequenzwerte berechnet.

Gestartet wird das Programm über Label C'. Ausgedruckt wird vom Rechner jeweils die aktuelle Länge und Frequenz, die Dämpfung in dB und die relative Phasenverschiebung in Grad.

Ein Rechenbeispiel

Als Rechenbeispiel wurde eine Anordnung gewählt, wie sie in der Praxis auch auftreten kann (*Bild 2*). Die Parallelschaltung eines Widerstandes R=1 MΩ und einer Kapazität C=40 pF repräsentiert den Eingangswiderstand eines Meßgeräts (z. B. eines Oszillographen). Die Leitung entspricht einem 2 m langen 60-Ω-Koaxialkabel. Dabei wurden die folgenden Leitungsparameter ermittelt: R' = 0, L' = 370 nH/m, G' = 0 und C' = 102 pF/m.

Um den Abschlußwiderstand $\underline{Z}_2$ zu ermitteln, schreiben wir für die Kapa-

zität 1/pC; für eine Induktivität müßte man pL schreiben. Aus der Formel für die Parallelschaltung zweier Widerstände

$$R = \frac{1}{\frac{1}{R_1} + \frac{1}{R_2}}$$

erhalten wir für unser Beispiel:

$$\underline{Z}_2 = \frac{1}{\frac{1}{R} + pC}$$

Nachdem der Rechner mit dem Programm (Speicher-Bereichsverteilung 639.39) geladen ist, können die Daten eingelesen werden. Der Grad des Zählerpolynoms n=0 wird eingegeben und Label A gedrückt, dann wird der Koeffizient A_0=1 eingegeben und die R/S-Taste gedrückt. Der Grad des Nennerpolynoms m=1 wird eingegeben, und anschließend auf die Taste Label B ge-

Tabelle:

Eingabeprozedur

Prozedur	Eingabe	Befehl
1. Eingabe des Zählergrades	n	A
2. Eingabe der Zählerkoeffizienten	$A_0...A_n$	R/S
3. Eingabe des Nennergrades	m	B
4. Eingabe der Nennerkoeffizienten	$B_0...B_m$	R/S
5. Eingabe des Widerstandsbelags	R'	C
6. Eingabe des Induktivitätsbelags	L'	R/S
7. Eingabe des Ableitungsbelags	G'	R/S
8. Eingabe des Kapazitätsbelags	C'	R/S
9. Eingabe der Startfrequenz	f_{min}	D
10. Eingabe der Endfrequenz	f_{max}	R/S
11. Eingabe der Startlänge	l_{min}	E
12. Eingabe der Endlänge	l_{max}	R/S
13. Eingabe der Frequenz-Schrittweite	Δf	A'
14. Eingabe der Längen-Schrittweite	Δl	R/S
15. Eingabe der Werte pro Dekade bei logarithmischem Maßstab	i/Dek.	B'
16. Eingabe des Anzeigeformats (Diese Eingabe erfolgt nur, wenn ein anderes Format als zwei Stellen nach dem Komma gewünscht wird)	0...9	R/S
17. (Neuer) Start		C'

000	05	5	048	16	A'	096	01	1	144	48	48	192	04	04	241	43	RCL	290	65	×
001	69	DP	049	36	PGM	097	94	+/-	145	49	PRD	193	12	B	242	40	40	291	43	RCL
002	17	17	050	04	04	098	49	PRD	146	49	49	194	60	DEG	243	42	STO	292	08	08
003	43	RCL	051	18	C'	099	40	40	147	43	RCL	195	32	X:T	244	01	01	293	85	+
004	27	27	052	42	STO	100	49	PRD	148	48	48	196	22	INV	245	43	RCL	294	01	1
005	91	R/S	053	42	42	101	41	41	149	42	STO	197	37	P/R	246	41	41	295	95	=
006	76	LBL	054	32	X:T	102	71	SBR	150	01	01	198	32	X:T	247	42	STO	296	22	INV
007	18	C'	055	42	STO	103	58	FIX	151	43	RCL	199	28	LOG	248	02	02	297	28	LOG
008	25	CLR	056	43	43	104	42	STO	152	49	49	200	65	×	249	36	PGM	298	42	STO
009	71	SBR	057	71	SBR	105	46	46	153	42	STO	201	02	2	250	05	05	299	05	05
010	38	SIN	058	35	1/X	106	85	+	154	02	02	202	00	0	251	17	B'	300	22	INV
011	01	1	059	36	PGM	107	43	RCL	155	43	RCL	203	95	=	252	92	RTN	301	59	INT
012	06	6	060	04	04	108	38	38	156	36	36	204	42	STO	253	76	LBL	302	85	+
013	42	STO	061	18	C'	109	95	=	157	36	PGM	205	46	46	254	38	SIN	303	43	RCL
014	00	00	062	36	PGM	110	42	STO	158	04	04	206	01	1	255	87	IFF	304	05	05
015	43	RCL	063	05	05	111	06	06	159	16	A'	207	03	3	256	02	02	305	95	=
016	23	23	064	14	D	112	32	X:T	160	43	RCL	208	69	DP	257	42	STO	306	59	INT
017	42	STO	065	42	STO	113	42	STO	161	37	37	209	04	04	258	02	2	307	65	×
018	05	05	066	36	36	114	47	47	162	36	PGM	210	58	FIX	259	65	×	308	43	RCL
019	71	SBR	067	32	X:T	115	85	+	163	04	04	211	40	IND	260	89	π	309	28	28
020	44	SUM	068	42	STO	116	43	RCL	164	16	A'	212	09	09	261	95	=	310	55	÷
021	43	RCL	069	37	37	117	39	39	165	36	PGM	213	43	RCL	262	22	INV	311	01	1
022	01	01	070	71	SBR	118	95	=	166	04	04	214	46	46	263	49	PRD	312	00	0
023	42	STO	071	35	1/X	119	42	STO	167	13	C	215	69	DP	264	31	31	313	95	=
024	38	38	072	36	PGM	120	07	07	168	43	RCL	216	06	06	265	87	IFF	314	42	STO
025	43	RCL	073	04	04	121	43	RCL	169	42	42	217	22	INV	266	00	00	315	31	31
026	02	02	074	13	C	122	38	38	170	36	PGM	218	58	FIX	267	98	ADV	316	76	LBL
027	42	STO	075	36	PGM	123	75	-	171	04	04	219	03	3	268	43	RCL	317	43	RCL
028	39	39	076	05	05	124	43	RCL	172	16	A'	220	03	3	269	30	30	318	43	RCL
029	01	1	077	14	D	125	46	46	173	43	RCL	221	02	2	270	32	X:T	319	31	31
030	00	0	078	42	STO	126	95	=	174	43	43	222	03	3	271	43	RCL	320	32	X:T
031	42	STO	079	40	40	127	42	STO	175	36	PGM	223	02	2	272	08	08	321	43	RCL
032	00	00	080	32	X:T	128	48	48	176	04	04	224	04	4	273	22	INV	322	34	34
033	43	RCL	081	42	STO	129	43	RCL	177	16	A'	225	69	DP	274	67	EQ	323	77	GE
034	22	22	082	41	41	130	39	39	178	36	PGM	226	04	04	275	48	EXC	324	34	ΓX
035	42	STO	083	43	RCL	131	75	-	179	04	04	227	32	X:T	276	25	CLR	325	61	GTO
036	05	05	084	32	32	132	43	RCL	180	18	C'	228	58	FIX	277	42	STO	326	23	LNX
037	71	SBR	085	49	PRD	133	47	47	181	43	RCL	229	40	IND	278	08	08	327	76	LBL
038	44	SUM	086	40	40	134	95	=	182	06	06	230	09	09	279	43	RCL	328	34	ΓX
039	43	RCL	087	49	PRD	135	42	STO	183	36	PGM	231	69	DP	280	31	31	329	98	ADV
040	38	38	088	41	41	136	49	49	184	04	04	232	06	06	281	42	STO	330	02	2
041	36	PGM	089	71	SBR	137	93	.	185	16	A'	233	22	INV	282	28	28	331	01	1
042	04	04	090	58	FIX	138	05	5	186	43	RCL	234	58	FIX	283	76	LBL	332	69	DP
043	16	A'	091	42	STO	139	49	PRD	187	07	07	235	42	STO	284	48	EXC	333	04	04
044	43	RCL	092	38	38	140	06	06	188	36	PGM	236	47	47	285	69	DP	334	32	X:T
045	39	39	093	32	X:T	141	49	PRD	189	04	04	237	61	GTO	286	28	28	335	69	DP
046	36	PGM	094	42	STO	142	07	07	190	16	A'	238	18	C'	287	43	RCL	336	06	06
047	04	04	095	39	39	143	49	PRD	191	36	PGM	239	76	LBL	288	30	30	337	65	×
												240	58	FIX	289	35	1/X	338	02	2

drückt. Die Koeffizienten des Nenners sind $B_0 = 1/R$ und $B_1 = C$. Setzt man die Werte der Bauelemente in Grundeinheiten ein, so ergibt sich $B_0 = 10^{-6}$ und $B_1 = 40 \cdot 10^{12}$. Wir lesen zuerst B_0 (= 1EE+/−6) ein und drücken die R/S-Taste, anschließend wird B_1(=40EE+/−12) eingegeben und wieder die R/S-Taste gedrückt.

Jetzt müssen die Leitungsdaten eingegeben werden. Als erster Wert wird R'(=0) eingegeben und die Taste C gedrückt. Dann folgt L' (=370EE+/−9) und R/S. Genauso wie L' wird G'(=0) eingegeben und als letzter Wert C'(=102EE+/−12).

Es müssen jetzt noch der Frequenzbereich und die Länge festgelegt werden. Als Startfrequenz dienen 10 MHz und 30 MHz als obere Grenze mit fünf Zwischenwerten pro Dekade (logarithmischer Maßstab). Zudem soll der Rechner das Problem jeweils für einen Meter und dann für zwei Meter berechnen.

```
     0. 00     N        25. 06     F
     1. 00              -11.76 00     A
     1. 00     M        10.98-03     PHI
     1. -06
    40. -12
     0. 00     R'        2. 00     L
   370. -09    L'
     0. 00     G'       10. 06     F
   102. -12    C'       -4.28 00     A
     1. 00     L         3.94-03     PHI

    10. 06     F        16. 06     F
 -1.22 00      A       -19.94 00     A
  1.49-03      PHI      32.37-03     PHI

    16. 06     F        25. 06     F
 -3.41 00      A        -3.03 00     A
  2.96-03      PHI     180.00 00     PHI
```

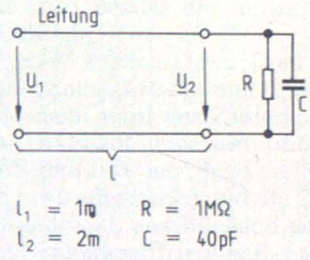


Bild 2. Schaltung des Rechenbeispiels und das Ein-/Ausgabeprotokoll

Die Startfrequenz (=10EE6) wird eingetippt und anschließend Label D betätigt, dann wird die Endfrequenz eingegeben (=30EE6) und R/S gedrückt. Die Anfangslänge des Kabels (=1) wird eingegeben und Label E gedrückt, die maximale Länge des Kabels (=2) wird eingegeben und im An-

```
339  65  ×      388  77  GE     437  30  30     486  32  32     535  42  STO    584  98  ADV    612  06  06
340  89  π      389  79  x̄      438  97  DSZ    487  91  R/S    536  00  00     585  91  R/S    613  02  2
341  95  =      390  98  ADV    439  05  05     488  42  STO    537  69  DP     586  76  LBL    614  02  2
342  42  STO    391  98  ADV    440  33  x²     489  33  33     538  25  25     587  13  C      615  06  6
343  31  31     392  98  ADV    441  92  RTN    490  42  STO    539  25  CLR    588  42  STO    616  05  5
344  92  RTN    393  91  R/S    442  76  LBL    491  35  35     540  76  LBL    589  24  24     617  69  DP
345  76  LBL    394  76  LBL    443  35  1/X    492  91  R/S    541  52  EE     590  03  3      618  04  04
346  42  STO    395  79  x̄      444  43  RCL    493  76  LBL    542  91  R/S    591  05  5      619  43  RCL
347  22  INV    396  86  STF    445  24  24     494  16  A'     543  99  PRT    592  06  6      620  25  25
348  86  STF    397  02  02     446  42  STO    495  86  STF    544  72  ST*    593  05  5      621  91  R/S
349  02  02     398  61  GTO    447  01  01     496  00  00     545  00  00     594  69  DP     622  42  STO
350  98  ADV    399  38  SIN    448  43  RCL    497  42  STO    546  69  DP     595  04  04     623  26  26
351  02  2      400  76  LBL    449  25  25     498  30  30     547  20  20     596  43  RCL    624  69  DP
352  07  7      401  44  SUM    450  65  ×      499  91  R/S    548  97  DSZ    597  24  24     625  06  06
353  69  DP     402  43  RCL    451  43  RCL    500  42  STO    549  05  05     598  69  DP     626  01  1
354  04  04     403  05  05     452  31  31     501  35  35     550  52  EE     599  06  06     627  05  5
355  43  RCL    404  44  SUM    453  95  =      502  91  R/S    551  98  ADV    600  02  2      628  06  6
356  32  32     405  00  00     454  42  STO    503  76  LBL    552  03  3      601  07  7      629  05  5
357  69  DP     406  69  DP     455  02  02     504  17  B'     553  00  0      602  06  6      630  69  DP
358  06  06     407  25  25     456  43  RCL    505  22  INV    554  69  DP     603  05  5      631  04  04
359  25  CLR    408  25  CLR    457  26  26     506  86  STF    555  04  04     604  69  DP     632  43  RCL
360  42  STO    409  42  STO    458  36  PGM    507  00  00     556  25  CLR    605  04  04     633  26  26
361  08  08     410  01  01     459  04  04     508  42  STO    557  91  R/S    606  43  RCL    634  91  R/S
362  43  RCL    411  42  STO    460  16  A'     509  30  30     558  76  LBL    607  24  24     635  42  STO
363  29  29     412  02  02     461  43  RCL    510  91  R/S    559  12  B      608  91  R/S    636  27  27
364  42  STO    413  76  LBL    462  27  27     511  42  STO    560  42  STO    609  42  STO    637  69  DP
365  28  28     414  33  X²     463  65  ×      512  09  09     561  23  23     610  25  25     638  06  06
366  32  X↕T    415  25  CLR    464  43  RCL    513  91  R/S    562  42  STO    611  69  DP     639  81  RST
367  61  GTO    416  36  PGM    465  31  31     514  76  LBL    563  05  05
368  34  fX     417  04  04     466  95  =      515  11  A      564  69  DP
369  76  LBL    418  16  A'     467  36  PGM    516  57  ENG    565  06  06
370  98  ADV    419  43  RCL    468  04  04     517  98  ADV    566  01  1
371  43  RCL    420  31  31     469  16  A'     518  42  STO    567  06  6
372  30  30     421  36  PGM    470  92  RTN    519  22  22     568  42  STO
373  44  SUM    422  04  04     471  76  LBL    520  42  STO    569  00  00
374  31  31     423  16  A'     472  14  D      521  05  05     570  69  DP
375  61  GTO    424  36  PGM    473  86  STF    522  03  3      571  25  25
376  43  RCL    425  04  04     474  02  02     523  01  1      572  25  CLR
377  76  LBL    426  13  C      475  42  STO    524  69  DP     573  76  LBL
378  23  LNX    427  85  +      476  28  28     525  04  04     574  57  ENG
379  43  RCL    428  73  RC*    477  42  STO    526  02  2      575  91  R/S
380  35  35     429  00  00     478  29  29     527  42  STO    576  99  PRT
381  44  SUM    430  95  =      479  91  R/S    528  09  09     577  72  ST*
382  32  32     431  42  STO    480  42  STO    529  43  RCL    578  00  00
383  43  RCL    432  01  01     481  34  34     530  22  22     579  69  DP
384  32  32     433  32  X↕T    482  91  R/S    531  69  DP     580  20  20
385  32  X↕T    434  42  STO    483  76  LBL    532  06  06     581  97  DSZ
386  43  RCL    435  02  02     484  15  E      533  01  1      582  05  05
387  33  33     436  69  DP     485  42  STO    534  00  0      583  57  ENG
```

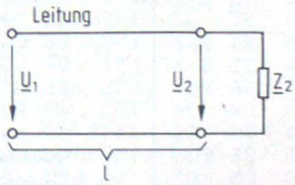


Bild 1. Programmlisting des ersten Programms mit dem Aufbau der Grundschaltung

```
000  05  5      080  32  X:T    160  13  C      240  58  FIX    320  32  X:T    400  76  LBL    440  33  X²
001  69  DP     081  42  STO    161  94  +/-    241  43  RCL    321  43  RCL    401  44  SUM    441  92  RTN
002  17  17     082  41  41     162  42  STO    242  40  40     322  34  34     402  43  RCL    442  76  LBL
003  43  RCL    083  43  RCL    163  44  44     243  42  STO    323  77  GE     403  05  05     443  35  1/X
004  27  27     084  32  32     164  32  X:T    244  01  01     324  34  FX     404  44  SUM    444  43  RCL
005  91  R/S    085  49  PRD    165  94  +/-    245  43  RCL    325  61  GTO    405  00  00     445  24  24
006  76  LBL    086  40  40     166  42  STO    246  41  41     326  23  LNX    406  69  DP     446  42  STO
007  18  C'     087  49  PRD    167  45  45     247  42  STO    327  76  LBL    407  25  25     447  01  01
008  25  CLR    088  41  41     168  43  RCL    248  02  02     328  34  FX     408  25  CLR    448  43  RCL
009  71  SBR    089  71  SBR    169  38  38     249  36  PGM    329  98  ADV    409  42  STO    449  25  25
010  38  SIN    090  58  FIX    170  44  SUM    250  05  05     330  02  2      410  01  01     450  65  ×
011  01  1      091  42  STO    171  01  01     251  17  B'     331  01  1      411  42  STO    451  43  RCL
012  06  6      092  38  38     172  44  SUM    252  92  RTN    332  69  DP     412  02  02     452  31  31
013  42  STO    093  32  X:T    173  44  44     253  76  LBL    333  04  04     413  76  LBL    453  95  =
014  00  00     094  42  STO    174  43  RCL    254  38  SIN    334  32  X:T    414  33  X²     454  42  STO
015  43  RCL    095  39  39     175  39  39     255  87  IFF    335  69  DP     415  25  CLR    455  02  02
016  23  23     096  01  1      176  44  SUM    256  02  02     336  06  06     416  36  PGM    456  43  RCL
017  42  STO    097  94  +/-    177  02  02     257  42  STO    337  65  ×      417  04  04     457  26  26
018  05  05     098  49  PRD    178  44  SUM    258  02  2      338  02  2      418  16  A'     458  36  PGM
019  71  SBR    099  40  40     179  45  45     259  65  ×      339  65  ×      419  43  RCL    459  04  04
020  44  SUM    100  49  PRD    180  43  RCL    260  89  π      340  89  π      420  31  31     460  16  A'
021  43  RCL    101  41  41     181  44  44     261  95  =      341  95  =      421  36  PGM    461  43  RCL
022  01  01     102  71  SBR    182  36  PGM    262  22  INV    342  42  STO    422  04  04     462  27  27
023  42  STO    103  58  FIX    183  04  04     263  49  PRD    343  31  31     423  16  A'     463  65  ×
024  38  38     104  42  STO    184  16  A'     264  31  31     344  92  RTN    424  36  PGM    464  43  RCL
025  43  RCL    105  46  46     185  43  RCL    265  87  IFF    345  76  LBL    425  04  04     465  31  31
026  02  02     106  32  X:T    186  45  45     266  00  00     346  42  STO    426  13  C      466  95  =
027  42  STO    107  42  STO    187  36  PGM    267  98  ADV    347  22  INV    427  85  +      467  36  PGM
028  39  39     108  47  47     188  04  04     268  43  RCL    348  86  STF    428  73  RC*    468  04  04
029  01  1      109  43  RCL    189  16  A'     269  30  30     349  02  02     429  00  00     469  16  A'
030  00  0      110  42  42     190  36  PGM    270  32  X:T    350  98  ADV    430  95  =      470  92  RTN
031  42  STO    111  75  -      191  04  04     271  43  RCL    351  02  2      431  42  STO    471  76  LBL
032  00  00     112  43  RCL    192  18  C'     272  08  08     352  07  7      432  01  01     472  14  D
033  43  RCL    113  36  36     193  43  RCL    273  22  INV    353  69  DP     433  32  X:T    473  86  STF
034  22  22     114  95  =      194  36  36     274  67  EQ     354  04  04     434  42  STO    474  02  02
035  42  STO    115  36  PGM    195  36  PGM    275  48  EXC    355  43  RCL    435  02  02     475  42  STO
036  05  05     116  04  04     196  04  04     276  25  CLR    356  32  32     436  69  DP     476  28  28
037  71  SBR    117  11  A      197  16  A'     277  42  STO    357  69  DP     437  30  30     477  42  STO
038  44  SUM    118  43  RCL    198  43  RCL    278  08  08     358  06  06     438  97  DSZ    478  29  29
039  43  RCL    119  43  43     199  37  37     279  43  RCL    359  25  CLR    439  05  05     479  91  R/S
040  38  38     120  75  -      200  36  PGM    280  31  31     360  42  STO
041  36  PGM    121  43  RCL    201  04  04     281  42  STO    361  08  08
042  04  04     122  37  37     202  16  A'     282  28  28     362  43  RCL
043  16  A'     123  95  =      203  36  PGM    283  76  LBL    363  29  29
044  43  RCL    124  36  PGM    204  04  04     284  48  EXC    364  42  STO
045  39  39     125  04  04     205  13  C      285  69  DP     365  28  28
046  36  PGM    126  11  A      206  42  STO    286  28  28     366  32  X:T
047  04  04     127  43  RCL    207  44  44     287  43  RCL    367  61  GTO
048  16  A'     128  42  42     208  03  3      288  30  30     368  34  FX
049  36  PGM    129  85  +      209  05  5      289  35  1/X    369  76  LBL
050  04  04     130  43  RCL    210  01  1      290  65  ×      370  98  ADV
051  18  C'     131  36  36     211  07  7      291  43  RCL    371  43  RCL
052  42  STO    132  95  =      212  69  DP     292  08  08     372  30  30
053  42  42     133  36  PGM    213  04  04     293  85  +      373  44  SUM
054  32  X:T    134  04  04     214  58  FIX    294  01  1      374  31  31
055  42  STO    135  16  A'     215  40  IND    295  95  =      375  61  GTO
056  43  43     136  43  RCL    216  09  09     296  22  INV    376  43  RCL
057  71  SBR    137  43  43     217  43  RCL    297  28  LOG    377  76  LBL
058  35  1/X    138  85  +      218  44  44     298  42  STO    378  23  LNX
059  36  PGM    139  43  RCL    219  69  DP     299  05  05     379  43  RCL
060  04  04     140  37  37     220  06  06     300  22  INV    380  35  35
061  18  C'     141  95  =      221  22  INV    301  59  INT    381  44  SUM
062  36  PGM    142  36  PGM    222  58  FIX    302  85  +      382  32  32
063  05  05     143  04  04     223  04  4      303  43  RCL    383  43  RCL
064  14  D      144  16  A'     224  04  4      304  05  05     384  32  32
065  42  STO    145  36  PGM    225  01  1      305  95  =      385  32  X:T
066  36  36     146  04  04     226  07  7      306  59  INT    386  43  RCL
067  32  X:T    147  18  C'     227  69  DP     307  65  ×      387  33  33
068  42  STO    148  43  RCL    228  04  04     308  43  RCL    388  77  GE
069  37  37     149  46  46     229  32  X:T    309  28  28     389  79  x̄
070  71  SBR    150  36  PGM    230  58  FIX    310  55  ÷      390  98  ADV
071  35  1/X    151  04  04     231  40  IND    311  01  1      391  98  ADV
072  36  PGM    152  16  A'     232  09  09     312  00  0      392  98  ADV
073  04  04     153  43  RCL    233  69  DP     313  95  =      393  91  R/S
074  13  C      154  47  47     234  06  06     314  42  STO    394  76  LBL
075  36  PGM    155  36  PGM    235  22  INV    315  31  31     395  79  x̄
076  05  05     156  04  04     236  58  FIX    316  76  LBL    396  86  STF
077  14  D      157  16  A'     237  61  GTO    317  43  RCL    397  02  02
078  42  STO    158  36  PGM    238  18  C'     318  43  RCL    398  61  GTO
079  40  40     159  04  04     239  76  LBL    319  31  31     399  38  SIN
```

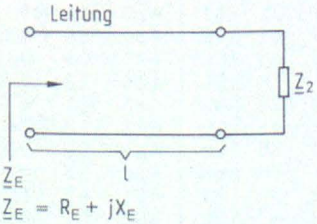


Bild 3. Programmlisting des Zusatzprogramms mit dem Aufbau der Grundschaltung

schluß daran die R/S-Taste gedrückt. Jetzt müßte die lineare Frequenzänderung eingegeben werden. obwohl wir dafür keinen Wert haben, muß trotzdem Label A' gedrückt werden, um die Schrittweite der Längenänderung über R/S eingeben zu können, nachdem wir ihren Wert (=1) eingelesen haben. Als letztes lesen wir ein, wieviel Werte pro Dekade wir berechnet haben wollen (= 5), und drücken dann die Taste von Label B'. Sollen die Frequenzwerte linear gewählt werden, darf die Taste Label B' nicht betätigt werden. Über die Taste Label C' wird das Programm gestartet.

Der Eingabemodus wird in der *Tabelle* noch einmal schematisiert dargestellt. Dabei taucht eine Eingabe auf, die bisher noch nicht erwähnt wurde: Nachdem man die Anzahl der zu berechnenden Werte pro Dekade über Label B' eingegeben hat, kann man durch die Eingabe einer Zahl zwischen 0 und 9 und durch Drücken der R/S-Taste das Anzeigeformat verändern, das mit zwei Stellen nach dem Komma vorgewählt wird, wenn hier keine Eingabe erfolgt. Will man ein anderes Anzeigeformat und ein lineares Frequenz-Raster wählen, muß man, nachdem man das Anzeigeformat geändert hat, über Label A' einen linearen Maßstab eingeben und das Programm dann starten. Es gilt immer der zuletzt eingegebene Maßstab.

Sollte die Anordnung noch einmal mit anderen Parametern berechnet werden, so ist dies ohne Probleme und beliebig oft möglich. Müssen aber Grunddaten verändert werden (n, m, $A_0 \ldots A_n$, $B_0 \ldots B_m$, R', L', G', C'), ist es notwendig, das Programm neu einzulesen.

Als Ergebnis druckt der Rechner die aktuelle Frequenz und die Länge aus (F, L). „A" steht im Ausdruck für die Dämpfung in dB und „PHI" für die relative Phasenverschiebung in Grad.

Interessant sind an dem Ergebnis von unserem Rechenbeispiel die negative Dämpfung, d. h. eine Spannungsüberhöhung am Leitungsende, und der Betrag dieser Dämpfung, denn –20 dB entspricht einer Spannungsüberhöhung um den Faktor zehn (bei 20,35 MHz ist die Dämpfung schon auf –73,97 dB angewachsen). Interessant ist aber auch die Phasendrehung um 180 Grad bei höheren Frequenzen. Das Rechenbeispiel zeigt, daß Meßgeräte, wie z. B. Oszillografen mit einer längeren Meßleitung, das Meßergebnis bei höheren Frequenzen wesentlich beeinflussen können.

Abschließend sei bemerkt, daß es nicht sinnvoll ist, die Endfrequenz beliebig hoch anzusetzen. Dielektrische Verluste und Skineffekt vergrößern R' und G'. Diese Leitungsdaten sind dann auch bei kurzen Leitungen nicht mehr zu vernachlässigen, außerdem werden sie frequenzabhängig.

Ein- und Ausgangswiderstand einer Leitung

Mit dem Zusatzprogramm kann der Eingangswiderstand einer Leitung mit einem komplexen Abschlußwiderstand berechnet werden. Die Formel für den Eingangswiderstand $\underline{Z}_E$ lautet:

$$\underline{Z}_E = \underline{Z}_W \frac{e^{\underline{\gamma}l} + \underline{A}e^{-\underline{\gamma}l}}{e^{\underline{\gamma}l} - \underline{A}e^{-\underline{\gamma}l}} \; ; \; \underline{A} = \frac{\underline{Z}_2 - \underline{Z}_W}{\underline{Z}_2 + \underline{Z}_W}$$

wobei die hier auftretenden Variablen bereits in der vorangehenden Programmbeschreibung besprochen wurden.

Das Zusatzprogramm (*Bild 3*) für den TI-59, das ebenfalls mit dem Standard-Software-Modul und dem Drukker PC-100 verwendet wird, kann in drei Variationen angewendet werden.

1. Das Zusatzprogramm wird mit der Speicher-Bereichsverteilung 559.49 in den Rechner eingegeben und auf zwei Magnetkartenseiten gespeichert. Die Grunddaten werden hier über das erste Programm eingegeben, es können eventuell Berechnungen mit dem ersten Programm durchgeführt werden, dann wird das Zusatzprogramm in den Rechner eingelesen, und es können die Parameter wie im ersten Programm eingegeben werden.
2. Das Zusatzprogramm wird wie oben eingelesen, dann wird die Speicher-Bereichsverteilung auf 639.39 geändert. Es kann jetzt die dritte Magnetkartenseite des ersten Programms eingelesen werden (Programmspeicher 480 bis 639). Die gesamte Dateneingabe, einschließlich der Grunddaten, kann jetzt wie im ersten Programm erfolgen.
3. Das Zusatzprogramm wird mit einer Speicher-Bereichsverteilung von 639.39 eingelesen, und auch die dritte Magnetkartenseite des ersten Programms wird noch einmal kopiert. Das Zusatzprogramm ist jetzt ein eigenständiges Rechnerprogramm. Die Dateneingabe ist identisch mit dem ersten Programm.

Noch ein Rechenbeispiel

Als Rechenbeispiel wurde eine sog. offene ¼-Leitung gewählt (*Bild 4*), d. h. der Abschlußwiderstand ist unendlich groß. Wir wählen ihn aus rechentechnischen Gründen endlich mit 10^{20} Ω. Die Leitungsdaten sind die gleichen wie beim vorhergehenden Rechenbeispiel. Als Länge wählen wir 2 m. Die Startfrequenz soll 10 MHz betragen, die Endfrequenz 30 MHz. Als Vorschub werden zehn Werte pro Dekade gewählt (also wieder ein logarithmischer Maßstab).

Wir geben zunächst den Grad des Zählerpolynoms n=0 ein und drücken Label A, dann wird der Zählerkoeffizient A_0 (=1EE20) eingelesen und die Taste R/S gedrückt. Jetzt wird der Nennergrad m=0 eingegeben und die Taste Label B gedrückt, anschließend geben wir den Nennerkoeffizienten B_0=1 ein und drücken die R/S-Taste. Die Eingabe der Leitungsdaten, der Start- und der Endfrequenz wurde bereits im ersten Programmteil beschrieben.

Es folgt nun die Eingabe der Leitungslänge. Die Startlänge (=2) wird eingegeben und Label E gedrückt, dann wird die Endlänge (=2) eingegeben und die Tastenfolge R/S, A', R/S gedrückt. Als letztes folgt die Eingabe der Anzahl der Werte, die pro Dekade berechnet werden sollen (=10). Jetzt drücken wir auf Label B' und starten das Programm, indem wir noch die Taste Label C' betätigen.

Ausgedruckt werden die aktuelle Länge und die aktuelle Frequenz sowie der Realteil des komplexen Eingangswiderstandes mit „RE" und dessen Imaginärteil (Blindwiderstand) mit „XE" in Ohm.

Die Zahlenwerte unseres Beispiels zeigen, daß bei etwa 20 MHz eine Serienresonanz vorliegt. Bei Frequenzen unter 20 MHz überwiegt kapazitives Verhalten (negatives Vorzeichen), bei 25 MHz dagegen induktives Verhalten.

```
                0. 00    N
              100. 18

                0. 00    M
                1. 00

                0. 00    R'
              370. -09   L'
                0. 00    G'
              102. -12   C'

                2. 00    L

               10. 06    F
           126. 96-12    RE
           -61. 87 00    XE

               13. 06    F
            85. 72-12    RE
           -38. 37 00    XE

               16. 06    F
           135. 66-12    RE
           -21. 01 00    XE

               20. 06    F
           117. 58-12    RE
            -1. 62 00    XE

               25. 06    F
           139. 14-12    RE
            22. 61 00    XE
```

Bild 4. Ein-/Ausgabeprotokoll des Rechenbeispiels für eine offene λ/4-Leitung

Kassettenverwaltung für den CBM

Mit dem Programm MENUE kann man eines von mehreren, auf einer Kassette gespeicherten Programmen schnell auffinden und laden. MENUE wird als erstes Programm auf der Kassette abgespeichert, danach die weiteren Programme. Deren Namen werden in Data-Statements im MENUE-Programm eingetragen. Um diese Programme nach dem Laden von MENUE durch schnellen Vorlauf finden zu können, werden auch die Vorlaufzeiten eingetragen, und zwar für jedes Programm die Zeit vom Anfang des Programms bis zum Ende des Bandes im schnellen Vorlauf. Die Zeitmessung ist mit den Programmzeilen 500–600 möglich.

Will man ein Programm auf dem Band laden, so lädt man zunächst MENUE und startet es. Dann wird das Inhaltsverzeichnis der Kassette auf dem Bildschirm gezeigt und die Nummer des gewünschten Programms erfragt. Nach Eingabe der Nummer gibt MENUE Anweisungen zur Bedienung des Recorders und hält nach schnellem Vorlauf kurz vor dem zu ladenden Programm an.

Die Aufnahme weiterer Programme ist auch später noch möglich, wenn noch Platz auf der Kassette ist. Man geht dazu im schnellen Vorlauf (s. o.) bis vor das letzte, bereits abgespeicherte Programm und setzt das Band mit VERIFY an das Ende dieses Programms. Danach kann das neue Programm aufgenommen werden. Titel und Vorlaufzeit müssen dann noch in das Programm MENUE eingetragen werden. Damit MENUE hierdurch nicht länger wird und das folgende Programm überschreibt, müssen vorher entsprechend viele DATA-Statements eingefügt sein, die man z. B. mit Sternchen füllt.

Die Vorlaufzeit läßt sich sehr genau einstellen. Nachdem sie gemessen und eingetragen ist, wird man die gemessene Zeit prüfen und ggf. korrigieren, indem man mit MENUE in schnellem Vorlauf bis vor das neu aufgenommene Programm spult und dann die Zeit bis zum Auffinden des Programms mißt. Als Richtwert für diese Zeit kann man zehn Sekunden nehmen. Das Verhältnis der Geschwindigkeiten im schnellen Vorlauf und beim normalen Abspielen liegt ungefähr bei 20.

Die Auflistung erklärt die Wirkung des Programms MENUE weitgehend. Für den praktischen Gebrauch wird man das Programm stark kürzen wollen. Die vorliegende Version ist für den CBM 3000 geschrieben. Für den CBM 2000 sind nur in drei Zeilen Änderungen erforderlich:

162 R = PEEK (241)-1
168 CS = 207 + R
172 AT = 525

Das Programm stellt auch selbständig fest, welcher Recorder benutzt wird, so daß jede Kassette auf beiden

```
100 REM *** MENUE ***
101 :
102 REM BEIM EINLESEN DES PROGAMMS
103 REM SOLLTEN BEMERKUNGEN UND UN-
104 REM NOETIGE LEERSTELLEN ENTFERNT
105 REM WERDEN. VARIABLENNAMEN SOLLTEN
106 REM AUF IHRE BEIDEN ANFANGSBUCH-
107 REM STABEN VERKUERZT WERDEN.
109 :
110 REM UEBERSCHRIFT:
111 U$="ARCHIV 5, KASSETTE 8, SEITE A
112 :
113 ZAHL=10: REM GESPEICHERTE FILES
114 :
115 REM KENNZEICHNUNG FILEART B/D/M:
116 REM B = BASICPROGRAMM
117 REM D = DATENFILE
118 REM M = MASCHINENSPRACHE-PGM
119 :
121 DATADISASSEMBLER 4.1 B,2320
122 DATAEDITOR 4.1       B,2060
123 DATAMERGE            M,1877
124 DATAREPEAT-FUNKTION  M,1791
125 DATAPSEUDO-RESET     M,1716
126 DATADUMP             M,1638
127 DATAARRAYDUMP        M,1559
128 DATATOOLKIT          M,1474
129 DATASCHACH           M,1255
130 DATARENUMBER         M,0693
131 DATA**************** D,0000
132 DATA**************** D,0000
133 DATA**************** D,0000
134 DATA**************** D,0000
135 DATA**************** D,0000
136 DATA**************** D,0000
137 DATA**************** D,0000
138 DATA**************** D,0000
139 DATA**************** D,0000
140 DATA**************** D,0000
159 :
160 REM WERTE DER VARIABLEN
161 :
162 R=PEEK(212)-1: REM AKTIVER REKORDER
163 :
164 CM=59411+ 45*R: REM CASSETTENMOTOR
165 ME=   53+154*R: REM MOTOR EIN
166 MA=   61+162*R: REM MOTOR AUS
167 :
168 CS=249+R  : REM CASSETTENSTEUERUNG
169 SE=52-51*R: REM STEUERUNG EIN
170 SA=0      : REM STEUERUNG AUS
171 :
172 AT=  158: REM REGISTER FUER ANZAHL
173 :        REM DER GEDRUECKTEN
174 :        REM TASTEN AM CBM
175 :
176 WAHR=255: REM FUER WAIT BEFEHL
177 :
178 D$="DRUECKE JETZT '"
179 F$="F.FWD"
180 S$="STOP"
181 A$="' AM REKORDER!"
182 :
200 PRINT: PRINT U$: PRINT
210 DIM N$(ZAHL), ZEIT(ZAHL)
220 :  FOR K=1 TO ZAHL: READ N$(K), ZEIT(K)
230 :  PRINT TAB(-(K<10)) K; N$(K): NEXT
240 INPUT "FILE-NUMMER"; K
250 IF K<1 OR K>ZAHL THEN 240
260 PRINT: PRINT K; N$(K)
270 IF PEEK(CS)=SA GOTO 300
280 :  PRINT D$S$A$: WAIT CS,WA,SE
290 :  PRINT"DANKE
300 PRINT D$F$A$: WAIT CM,MA,MA
310 T=TI+ZEIT(1)-ZEIT(K): PRINT "DANKE
320 IF TI<T THEN 320
330 POKE CS,SE: POKE CM,MA
340 PRINT D$S$A$: WAIT CS,WA,SE
350 PRINT "UND NUN LADEN.
360 END
497 :
498 REM ZEITMESSUNG
499 :
500 PRINT D$F$A$: PRINT "UND SOBALD";
510 PRINT " DAS BAND GANZ DURCHGESPULT
520 PRINT "IST, DIE LEERTASTE.
530 WAIT CM,MA,MA: T=TI
540 PRINT "OK. - ZEIT LAEUFT.
550 POKE AT,0: WAIT AT,1
560 T=TI-T: PRINT T
READY.
```

Jeweils ein Programm auf einer Kassette abzuspeichern ist sicher auch eine Methode, Ordnung zu halten – dieses MENUE-Programm stellt aber eine wesentlich wirtschaftlichere Lösung dar. Es wird jeweils am Anfang jeder Kassette aufgezeichnet

Recordern ohne Änderungen geladen werden kann.

Schließlich noch eine Bemerkung zum WAIT-Befehl: In Zeile 320 wartet WAIT CS, WA, SE bis der Inhalt von CS (Kassettensteuerung) von SE (Steuerung ein) verschieden ist, also den Wert SA (Steuerung aus) annimmt. Die Form WAIT CS, SA wartet zwar i. a. bis (CS) AND SA von Null verschieden ist, sie kann aber hier wegen SA = 0 nicht benutzt werden. Übrigens hat WAIT A, B für A = 6502 die überraschende Wirkung, daß B-mal MICROSOFT! auf den Bildschirm geschrieben wird. Dr. Harald Lindner

Zusätzliche Speicherplätze beim SR-52

Bekanntlich hat der SR-52 nur 20 Speicherplätze (von R 00 bis R 19); dazu kommen noch 224 Programmschritte – so steht es wenigstens im SR-52-Handbuch. Um so überraschter war der Verfasser eines Tages, als er zufällig in einem seiner Programme die Befehle STO 90 und ein Stück weiter RCL 90 vorfand; ein Programmierfehler, denn es hätte STO 09 bzw. RCL 09 heißen sollen. Erstaunlicherweise funktionierte das Programm trotzdem!

Weitere Nachforschungen ergaben folgende „neue" Eigenschaften: Der SR-52 akzeptiert alle im Handbuch beschriebenen Speicherfunktionen (STO.., RCL.., EXC.., SUM.., IND STO.. usw.) auch mit den Adressen 70 bis 99, d. h. es existieren zusätzlich zu den bekannten Speichern 00 bis 19 noch weitere 30 Speicher. Davon werden R 70 bis R 97 innerhalb und R 98 sowie R 99 außerhalb des Programmspeichers abgelegt. Diese 30 Speicher sind mit CMs (= Clear Memories) *nicht* löschbar

Sichtbar werden die Speicherinhalte – außer mit RCL... – durch Umschalten des Rechners auf LRN, und zwar nimmt R 70 die Plätze 000 bis 007, R 97 die Plätze 216 bis 223 ein.

STO 97 ergibt mit x=1.234567654321 E 78 folgende Programmspeicherbelegung:

223	222	221	220	219	218	217	216	= Speicherplatz
12	34	56	76	54	32	17	80	= Speicherinhalt
			Mantisse			Exp.		

Know-how unter Wert verkauft

Die Mikrocomputer-Welle rollt. Immer mehr und immer kleinere Unternehmen entdecken, daß sie Produktions- und Verwaltungsabläufe mit Computern rationalisieren können. Gleichzeitig fallen die anfangs oft drastisch überhöhten Hardware-Preise. Dies bedeutet aber auch: Hersteller und Händler können nicht länger soviel Aufwand bei der Individualberatung des Einzelkunden treiben, wie dies im Zeitalter der Hardware-Mondpreise wirtschaftlich durchaus noch vertretbar war. Der Bedarf an anwenderspezifischer Software steigt und steigt. Professionelle Dienstleistung auf diesem Gebiet durch Softwarehäuser oder beratende Ingenieurbüros ist oftmals schwierig zu bekommen und dann in aller Regel sehr teuer: Stundensätze über 180 Mark wurden bezahlt! In dieser Situation entsinnen sich mehr und mehr (durchweg kleinere) Anwender der „Halbprofis" und Hobbyisten.

Von einem Generalimporteur für ein ausländisches Computersystem weiß man, daß seine „Software-Abteilung" aus einer Handvoll stundenweise entlohnter Studenten einer benachbarten Fachhochschule besteht. In einem rheinischen Chemie-Werkchen steuert ein PET den Ablauf gewisser Routine-Laboruntersuchungen und führt darüber Protokoll. Das Programm stammt von einem Hobbyisten, den der Firmenchef per Zufall kennenlernte. Ein mittelständischer Werbeverlag am Neckar fertigt Adressenaufkleber mit Hilfe eines ITT-2020. Autor des Programms: Ein Stuttgarter Gymnasiast.

Was ist von derlei Programmiererei im Nebenjob zu halten? Profis, wie zum Beispiel beratende Ingenieure, rümpfen die Nase, wenn sie davon hören, und sprechen dann gern von „Schwarzarbeit". Es ist nicht immer Futterneid, was solche unfreundlichen Kommentare hervorruft. Wie man hört, haben durch Dilettanten gefertigte Programme in einigen Fällen durchaus zu bösen Überraschungen geführt: Etwa dann, wenn sie gewartet werden sollten, der Autor längst entschwunden war und sein Nachfolger einen schier unentwirrbaren Statement-Salat vorfand. Die Chance, derartige Entdeckungen machen zu müssen, ist bei Dilettanten-Arbeit allemal höher als bei Profi-Programmen.

Auf der anderen Seite sind ein abgeschlossenes Informatik-Studium und jahrelange Erfahrung als Softwaremann auf Anlagen der mittleren Datentechnik durchaus kein Garantieschein dafür, daß der Mann optimal mit den Möglichkeiten eines Mikro-Computers umgehen kann. Ein gewissenhafter Hobbyist, der sein Maschinchen durch und durch kennt, mag hier dem Profi oftmals überlegen sein.

Was nun das böse Wort „Schwarzarbeit" angeht, so ist es schlicht als eine Art von Greuelpropaganda einzustufen. Rechtlich bestehen gegen die Programmentwicklung im Nebenjob nämlich keinerlei Hindernisse, solange sie als „freiberufliche wissenschaftliche Tätigkeit" erfolgt und die Einnahmen ordnungsgemäß versteuert werden.

Als problematisch erscheint hingegen, daß sich viele Nichtprofis bei der Annahme derartiger Aufträge aus Unkenntnis oder Bescheidenheit oft kräftig übers Ohr hauen lassen. Während der Unternehmer mit Hilfe der betreffenden Software vielstellige Summen pro Jahr einspart, geben sich viele Nebenjob-Programmierer mit dem Stundenlohn einer Raumpflegerin oder weniger zufrieden: Weil sie nicht wissen, daß auch ein Stundensatz von beispielsweise 50 Mark noch ein gutes Geschäft für den Auftraggeber wäre – routinierte und schnelle Qualitätsarbeit natürlich vorausgesetzt.

Solange die Nachfrage nach Programmierleistung gegenüber dem Angebot noch so deutlich überwiegt wie in diesen Tagen, sollten sich Nebenjob-Programmierer davor hüten, ihr Know-how unter Wert zu verkaufen. In Stuttgart haben sich unlängst einige dergestalt tätige nebenamtliche Programmierer getroffen und beschlossen, ihre Stundensätze abzustimmen und in etwa an jenen Summen zu orientieren, wie sie heutzutage von Auto-Reparaturwerkstätten in Rechnung gebracht werden. Wie man hört, seien ein paar Auftraggeber daraufhin abgesprungen, weil diese offenbar weiterhin Geschäfte auf der Basis „zehn Stunden Arbeit gegen einen Karton Disketten" vorzogen, aber, so heißt es dazu weiterhin, durch die Mehreinnahmen aufgrund der neuen Regelung seien diese Auftragsverluste bereits nach wenigen Wochen ausgeglichen gewesen. Ein Beispiel, das zu denken geben sollte.

Hans-Georg Joepgen

Die letzte Ziffer (hier 0) enthält die Information über die Vorzeichen von Mantisse und Exponent:

Mant.	Exp.	Ziffer
+	+	0
−	+	2
+	−	4
−	−	6

Belegter Programmspeicherplatz in Abhängigkeit von der Speicher-Nr. XX: 1. Platz = (XX − 70) · 8, letzter Platz = (XX − 70) · 8 + 7

Für das Arbeiten mit dem SR-52 hat dies alles einige praktische Folgen, die nicht im Handbuch beschrieben sind:

Es ist eine Datenspeicheraufteilung in Bereiche möglich, die mit CMs löschbar bzw. nicht löschbar sind. Beispiel: Bei der Berechnung einer linearen Regression werden die dazu nötigen Summen in den Speichern R 01 bis R 06 gebildet, die Ergebnisse in R 97 bis R 99 gespeichert. Für eine neue Rechnung werden mit CMs die Speicher R 01 bis R 06 gelöscht, während die Ergebnisse der neuen Rechnung mit denen der vorherigen Rechnung in R 97 bis R 99 verglichen werden können. Daten können zusammen mit Programmen auf Magnetkarten gespeichert bzw. geladen werden. Beispiel: Ein Statistik-Programm für die Normalverteilung enthält 12 Konstanten mit zusammen 107 Stellen. Einschließlich der STO-XX-Befehle ergibt dies 143 Tastenbetätigungen von Hand oder – im Programm untergebracht – 143 Programmschritte. Für das umfangreiche Programm selbst bleibt da nicht mehr viel Platz übrig! Speichert man dagegen diese 12 Konstanten in den Registern R 86 bis R 97, werden nur 96 Programmspeicherplätze belegt.

Die Speicherung von vielstelligen Konstanten in Registern und deren Aufruf während der Rechnung beeinflussen besonders bei Iterationsverfahren auch die Rechengeschwindigkeit: RCL 90 STO 01 benötigt ca. 120 Millisekunden;
1.234567890 STO 01 dagegen ca. 180 Millisekunden.

Vorsicht ist jedoch beim Programmieren geboten: Einfügen (INS) und Löschen (DEL) von Programmschritten verschieben natürlich auch die Konstanten bzw. Daten in R 70 bis R 97; R 98 und R 99 bleiben dagegen unbeeinflußt.

Rolf Englert

```
0200 A9 54 8D 12
0204 A4 A9 00 8D
0208 34 A4 20 F0
020C E9 A2 00 20
0210 46 02 A9 00
0214 8D 15 01 20
0218 53 ED CA 8E
021C 15 A4 BD 16
0220 01 D0 EF E8
0224 BD 16 01 20
0228 7A E9 E8 E0
022C 06 D0 F5 BD
0230 16 01 C9 0D
0234 D0 08 A2 17
0238 20 46 02 4C
023C 12 02 A2 12
0240 20 46 02 4C
0244 12 02 BD 55
0248 02 F0 06 20
024C 7A E9 E8 D0
0250 F5 20 F0 E9
0254 60 43 41 53
0258 53 45 54 54
025C 45 20 43 4F
0260 4E 54 45 4E
0264 54 53 00 2E
0268 54 58 54 00
026C 2E 4F 42 4A
0270 00 20 41 55
010C 4C 00 02 5A
```

Bild 2. Hexadezimales Programmlisting für das Auflisten von Kassetteninhalten

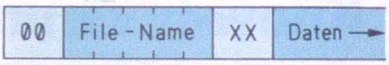


Bild 3. Anfang des „Tape-Buffers" nach dem Einlesen des ersten Blocks einer File. „XX" enthält bei Objekt-Files ein CR-Zeichen (hex 0D)

Kassetten-Listing mit AIM-65

Das hier beschriebene Programm erstellt eine Liste der auf einer Kassette gespeicherten Files, die im AIM-65-Text- oder Objektformat vorliegen (vgl. User Manual Appendix F). Zusätzlich zum File-Namen druckt das Programm den Suffix „.OBJ", wenn ein Objekt-File gefunden wurde, und „.TXT" bei Basic- oder Source-Files. Für die Vergabe von File-Namen bedeutet das eine Erleichterung: Source- und Objekt-Code dürfen den gleichen Namen haben. *Bild 1* zeigt einen Musterausdruck.

Verfolgt man während des Auslistens das Zählwerk des Kassetten-Recorders, so kann man den Beginn des jeweiligen Files festhalten. Der File-Name wird gedruckt, nachdem der erste Block eingelesen wurde; deshalb liegt der physikalische Anfang etwa einige Zählwerkeinheiten davor. Damit kann z. B. ein neues Verzeichnis geschrieben werden, wenn ein Recorder mit anderem Zählwerk verwendet werden soll.

Nicht lesbare Blöcke führen zu einer unkommentierten Fehlermeldung, Bandstop und Rücksprung zum Monitor. Durch Drücken der Taste „F1" nimmt das Programm nach der fehlerhaften Stelle seine Arbeit wieder auf – wie beim ursprünglichen Start.

Bild 2 enthält das Listing des Programms. Es wird mit der F1-Taste gestartet, lädt „T" als Input Device in INFLG, Tape Unit 1 in TAPIN und druckt den Kopf des Listings. In Anlehnung an die Monitor-Routine LOADTA ($E32F) wird ein Block von der Kassette in den Tape-Buffer gelesen. Dies wiederholt sich so lange, bis der Anfang eines Files gefunden wurde (Block Nr. 0). Den Beginn des Tape-Buffers nach Einlesen von Block 0 zeigt *Bild 3*.

Nachdem der File-Name ausgedruckt ist, prüft das Programm das auf den Namen folgende Byte. Es enthält $0D („CR"), wenn es sich um einen Objekt-File handelt.

```
<E>
CASSETTE CONTENTS
HIPAS.TXT
LOPAS.TXT
MUSI1.OBJ
BACH1.OBJ
PLOT .TXT
PLOT .OBJ
PLOTA.TXT
PLOTB.TXT
PLOTS.TXT
SYMTB.TXT
SYMTB.OBJ
EPROM.TXT
EPROM.OBJ
SYSRT.TXT
SYSRT.OBJ
PLOTD.OBJ
DIRY .TXT
DIRY .OBJ
```

Bild 1. Beispielausdruck mit dem AIM-65: Inhalt einer Kassette. Der Start erfolgt mit Taste F1, abgebrochen wird mit Reset

Nachdem das Listing-Programm geladen wurde, legt man die zu prüfende Kassette in den Recorder 1, spult sie zurück und stellt das Zählwerk auf Null. Der Drucker wird eingeschaltet und das Programm mit der F1-Taste gestartet, anschließend der Recorder auf Wiedergabe geschaltet. Jeweils beim Ausdruck eines File-Namens kann der Zählwerkstand notiert werden. Die Nummer des gerade eingelesenen Blockes wird vom Display angezeigt; so läßt sich auch die Länge der Files festhalten. Nach einem nicht lesbaren Block bleibt die Kassette mit einer Fehlermeldung „ERROR" stehen; das Listing kann anschließend mit F1 weitergeführt werden. Mit der Reset-Taste wird es beendet. (Anm. d. Red.: Objektprogramme, die mit dem in [1] veröffentlichten Kopierprogramm dupliziert wurden, werden fälschlich mit „TXT" ausgegeben. Dies liegt am Kopier-, nicht am Auflist-Programm. Ein Programm, das die gespeicherten Adressenbereiche von Objekt-Files auflistet, findet sich in [2].)

Dipl.-Ing. Georg J. Praml

Literatur

[1] AIM kopiert Kassetten. FUNKSCHAU 1980, Heft 16, S. 90.

[2] AIM-Directory. Sonderheft „Programme für Kleincomputer und Taschenrechner". Franzis-Verlag, München.

Fritz-B. Raab

Entfernungsliste für Basic-Rechner

Eine Entfernungstabelle zu besitzen ist für viele Funkamateure, die sich mit VHF-UHF-SHF-Bändern beschäftigen, wünschenswert. Für die Funkamateure und Computerfans, die einen Mikrocomputer besitzen und ein paar QTH-Kenner-Großfelder für den nächsten BBT oder Aktivitätsabend rechnen wollen, hat der Verfasser versucht, ein möglichst für alle Basic-Versionen taugliches Druckprogramm zu schreiben, das obendrein einfach auf jede Art von Drucker zugeschnitten werden kann.

Der Verfasser besitzt einen TRS-80 mit einem Matrix-Drucker OKI ET 5200. Das Programm ist jedoch so ausgelegt, daß es auch für einen Rechner mit nur 4 K RAM und einem alten Fernschreiber als Drucker arbeitet. Die Software für den TTY-Driver ist jedoch nicht im Programm enthalten, da dies sehr hinderlich für den Umbau auf andere Basic-Versionen wäre. Einschlägige Literatur zu diesem Thema ist schon genügend veröffentlicht worden, z. B. in FUNKSCHAU-Sonderheften.

Vom Ablauf her ist das Programm äußerst komfortabel. Der Benutzer braucht nur seinen eigenen QTH-Kenner einzugeben. Im Dialog will der Rechner dann die diagonalen „Eckfelder" wissen, zwischen denen er dann sämtliche Großfelder berechnet. Das bedeutet, daß das südwestlichste und das nordöstlichste Feld eines größeren Abschnittes anzugeben ist. Will man

```
10 REM QTH-KENNER-LISTE DRUCKEN OKTOBER 1980
20 REM DIESE VERSION EIGNET SICH F. ALLE DRUCKER UND TELETYPES
30 REM WER EINEN OKI DRUCKER HAT SOLLTE ALLE '*** IM PROGRAMM
40 REM ENTFERNEN . DANN WIRD DAS PAPIER OPTIMAL GENUETZT !!!
50 CLEAR 200:DIMY(2),B$(10),E(20)
60 CLS:PRINTCHR$(23):PRINT@450,"QTH-KENNER-LISTE"
70 PRINT@796,"DF2RF OKTOBER 1980"
80 GOSUB 1000
90 PRINT"GEBEN SIE DEN QTH-KENNER AN, FUER DEN GERECHNET WERDEN
SOLL!":INPUTQ$:I=1
100 PRINT:PRINT:PRINT:PRINT"GEBEN SIE DIE ECKFELDER AN, ZWISCHEN
 DENEN GERECHNET WERDEN     SOLL ! (1.SUEDWESTLICHSTES; 2.
NORDOESTLICHSTES)":INPUTA$(1),A$(2)
110 XX=ASC(A$(1)):XY=ASC(A$(2))
120 UU=ASC(RIGHT$(A$(1),1)):UY=ASC(RIGHT$(A$(2),1))
130 B$=CHR$(XX)+CHR$(UU):GOTO170
140 IFUU=UYANDXX=XYTHENEND ELSE IFUU(UYANDXX(=XYTHENUU=UU+1:GOTO160
150 IFUU=UYANDXX(XYTHENUU=ASC(RIGHT$(A$(1),1)):XX=XX+1
160 B$=CHR$(XX)+CHR$(UU):GOTO170
170 LS=(ASC(B$)-65)*2*3600:BS=(ASC(RIGHT$(B$,1))-24)*3600
180 IFASC(B$))85THEN LS=(ASC(B$)-92)*7200
190 PRINT"FUER FELD ";B$;" WURDEN FOLGENDE WERTE ERRECHNET :":PRINT
"LAENGE IN SEKUNDEN :";LS;"   BREITE IN SEKUNDEN :";BS
200 DEFINT K,O,P,T,W,Y:G$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
210 E$="3555311135531113 53"
220 FORJ=1TO2
230 GOSUB310
240 IFK)21THENK=K-26
250 Y(J)=K:NEXTJ
260 L=(2*Y(1))-2:B=Y(2)+39:X=VAL(MID$(Q$,3,2))-1:D=INT(X/10)
270 L=L+(X-10*D)/5:B=B+(7-D)/8:J=5:GOSUB310
280 IFK=10THENK=9
290 L=L+(VAL(MID$(E$,K,1)))/30:B=B+(VAL(MID$(E$,K+9,1)))/48
300 L1=L*3600:B1=B*3600:GOTO360
310 X$=MID$(Q$,J,1)
320 FORK=1TO36
330 IFX$=MID$(G$,K,1)THENGOTO350
340 NEXTK
350 RETURN
360 PRINT
370 PRINTQ$;" IST DER AUSGANGSPUNKT. IN SEKUNDEN:"
380 PRINT"LAENGE:";L1;"   BREITE:";B1
390 GOSUB 400
400 R=206369:O=1:T=1:W=1:P=0
410 FORI=1TO5:LPRINT" ":NEXTI
420 '*** LPRINTCHR$(27);CHR$(66);CHR$(29)
430 LPRINTTAB(15)"ENTFERNUNGEN VOM QTH-KENNER ";Q$;
" IN DAS GROSSFELD ";B$
440 FORI=1TO2:LPRINT" ":NEXTI
450 LPRINT"      A   B   C   D   E   F   G   H   J";
'***:LPRINT"          A   B   C   D   E   F
G   H   J"
460 '***Y=1:GOTO480
470 '***Y=2
480 IFO=1THEN530
490 IFO(=4THEN570
500 IFO=5THEN530
510 IFO(=8THEN550
520 IFO=9THEN530
530 K=360
540 GOTO580
550 K=120
560 GOTO580
570 K=600
580 G2=(LS+((T-1)*720+K))/R
590 G1=L1/R
600 IFG1(G2THEN630
610 G3=G1-G2
620 GOTO640
630 G3=G2-G1
640 A1=B1/R
650 IFO=3THEN740
660 IFO(=2THEN720
670 IFO(=6THEN760
680 IFO=7THEN740
690 IFO=8THEN720
700 IFO=9THEN740
710 GOTO530
720 K=75
730 GOTO770
740 K=225
750 GOTO770
760 K=375
770 A2=(BS-(P*450+K))/R
780 M=COS(A1)*COS(A2)*COS(G3)+SIN(A1)*SIN(A2)
790 S=ATN(SQR(1-M+2))*6368
800 Q=INT(S+.5)
810 IFY=2THENE(O+10)=Q:GOTO830
820 E(O)=Q
830 IFO=9THEN860
840 O=O+1
850 GOTO480
860 '***IFY=1THENP=P+4:O=1:GOTO470
870 '***IFW=21THENLPRINT" "
880 LPRINTUSING"##*";W;:FORI=1TO8:LPRINTUSING" ####";E(I);:NEXTI:
LPRINTUSING" ####";E(9)'***;:LPRINT"          ";USING"##*";
W+40;:FORI=1TO8:LPRINTUSING" ####";E(I+10);:NEXTI:
LPRINT USING " ####";E(19):IFW=40THEN1030ELSE W=W+1
890 IFW=80THEN990ELSEW=W+1'BEI OKI-DRUCKER DIESE ZEILE KILLEN
900 '***P=P-4:O=1
910 IFT(10THEN960
920 T=1
930 P=P+1
940 O=1
950 GOTO460
960 T=T+1
970 O=1
980 GOTO940
990 FORI=1TO5:LPRINT" ":NEXTI:LPRINTTAB(10) STRING$(45,"="):GOTO140
1000 FORA=1TO2000:NEXTA:CLS:PRINTCHR$(28):RETURN
```

Dieses Basic-Programm druckt eine Entfernungsliste für die bei Funkamateuren für die Standortangabe üblichen QTH-Kenner. Es wurde auf einem TRS-80 entwickelt, eignet sich nach Syntax-Anpassungen aber auch für andere Computer

nur ein Feld berechnen, so gibt man eben zweimal das gleiche an.

Über den Nullmeridian zu rechnen, d. h. Felder wie ZI, YH o. ä. zu berechnen, ist auch kein Hindernis, man muß jedoch bei einem größeren Lauf am Meridian aufhören und dann neu starten, da die Routine, welche die Großfelder zählt, dafür nicht vorgesehen ist.

Außerdem: Dies ist ein Basic-Programm! Ein Großfeld benötigt rund acht Minuten Rechenzeit. Immerhin werden in 12 Sekunden 18 Entfernungen berechnet! Alle Großfelder im Umkreis von 1000 Kilometer zu drucken dauert also ca. 26 Stunden (mit einem Fernschreiber entsprechend länger). Diese Arbeit wird man also ohnehin nicht in einem Stück machen. Wer zufällig einen TRS-80 mit Disk besitzt, kann gegen Einsendung einer Leerdiskette und Rückporto an den Verfasser eine compilierte Version des Programms erhalten (⅓ Rechenzeit).

Die Zeilen nach '*** sind für eine kompakte und schöne Druckformatierung, besonders für den OKI-Drucker gedacht. Für andere Drucker, die nicht die enge Schrift haben, sind je nach Druckersoftware leichte Änderungen nötig. Der Nachteil des relativ langsamen Druckes wird aufgewogen durch die gute Rechengenauigkeit. Die Rechenroutine selbst arbeitet mit Gradsekunden, also Integer-Zahlen, die keine Rundungen benötigen. Nur der Endwert wird dann sinnvollerweise auf ganze Kilometer gerundet. Die absolute Genauigkeit hängt zwar vom Computer ab, von der Anzahl der berücksichtigten Nachkommastellen, ist aber für die ohnehin ungenaue Angabe eines QTH-Kenners völlig ausreichend. In manchen Versionen des TRS-80 treten in der 7. Stelle hinter dem Komma (!) Rechenungenauigkeiten auf, die manchmal in Zeile 790 zum Error führen. Abhilfe: Einfügen einer Zeile 785 : IF 1-M ↑ 2<E-7 THEN M=1.

TI-59 berechnet Entfernungen aus QTH-Kennern

Im FUNKSCHAU-Sonderheft „Mikrocomputer-Anwendungen" beschreibt Laszlo Futo (HG4YR), wie man den programierbaren Taschenrechner TI-59 dazu verwenden kann, um mit den bei Funkamateuren üblichen QTH-Kennern als Ortsangaben Entferungen auszurechnen.

Das Kernproblem besteht darin, den QTH-Kenner-Code in die entsprechenden Längen- und Breitenangaben umzurechnen. Ausgegangen wird von den Großfeldern, die durch ganzzahlige Längen- und Breitengrade begrenzt werden. Vom linken unteren Eckpunkt des Großfeldes aus werden dann je nach Einzelfeld Gradanteile nach folgenden Formeln hinzuaddiert (Z1 = erste, Z2 = zweite Zahl des Einzelfeldes):

(Z2 – 1) x 0,2 = SUM 05 (Längengrad)
(7 – Z1) x 0,125 = SUM 06 (Breitengrad)

Diese Formeln liefern aber falsche Angaben, sobald Z2 Null ist. Die gesamte letzte Längsspalte des Großfeldes wird damit falsch berechnet.

Deshalb muß, falls Z2 Null ist, ein neues Label eingeschoben werden (nach Programmschritt 151: CP/RCL 08/EQ/PRT). Dieses Label vermindert dann Z1 um 1 und erhöht Z2 um 10. Da der Autor Laszlo Futo die Rechnerkapazität bereits voll ausgenützt hatte, muß im Programm eine Änderung der Speicherbereichs-Verteilung nach FUNKSCHAU 10/1980 erfolgen.

Thomas Denzler, Erwin Rathgeb

Acht Seiten machen sich selbständig

Wenn Sie, verehrter FUNKSCHAU-Leser, in zwei Wochen das nächste Heft aufschlagen, so werden Sie sich vermutlich wundern, daß plötzlich eine ganze Rubrik „fehlt": Der Mikrocomputer-Teil, in diesem Heft noch wie gewohnt zwischen den Rubriken „Elektroakustik mit HiFi" und „Praxis und Hobby" gelagert, macht sich nämlich selbständig und verschwindet damit aus der FUNKSCHAU.

Dies bringt zwei Vorteile für Sie mit sich. Erstens: Diejenigen unter Ihnen, die sich primär der Analog- oder der „konventionellen" Digitaltechnik verschrieben haben, brauchen nun nicht mehr acht Seiten „Ballast" in Kauf zu nehmen. Zweitens: Jene Leser, die sich primär für die Mikrocomputer-Technik interessieren, bekommen nun eine speziell für sie gemachte Publikation, die zunächst in zweimonatlichem Abstand erscheint.

Diese neue Publikation des Franzis-Verlages heißt – wie könnte es anders sein? – mc und erscheint erstmals am 27. April, also in rund einer Woche, im gewohnten Stil der FUNKSCHAU-Mikrocomputer-Rubrik: Grundlagenbeiträge, Hardware-Tips, Tests, Anwenderprogramme, Betriebssystem-Software, aktuelle Information und Produktneuheiten unterstützen den Mikrocomputer-Anwender beim Kauf, bei der Systemerweiterung und bei der Programmierung. Natürlich sind auch ein paar besondere Leckerbissen dabei – so etwa Programme, die im Strichcode abgedruckt sind, oder ein Beitrag über das Innenleben des Computers TRS-80 (mit 500 ROM- und RAM-Adressen!). In Heft 2 von mc finden Sie dann eine Bauanleitung für einen kleinen 3-Chip-Mikrocomputer mit der CPU 6504, der für beliebige Steuerungsaufgaben eingesetzt werden kann und nur rund 100 DM kostet.

Obwohl natürlich die FUNKSCHAU auch in Zukunft Neuheiten auf dem Mikrocomputer-Sektor nicht gerade verschweigen wird, können wir uns vorstellen, daß viele von Ihnen nun enttäuscht sind, weil sie die FUNKSCHAU vor allem wegen der Mikrocomputer-Rubrik abonniert haben. Damit das keine „Härtefälle" werden, erhalten alle Leser, die die FUNKSCHAU bis Ende 1981 schon bestellt haben, das mc-Abo 1981 für nur 16 DM. Ist das ein Angebot?

Ihr
Herwig Feichtinger

PET tilgt Raten

Das Programm in *Bild 1* errechnet für ein bestimmtes Kapital bei gleichbleibender Tilgung die jährlichen Zinsen, die jährliche Belastung (Tilgung + Zinsen) und die monatliche Belastung. Vom Kapital wird jährlich die gleichbleibende Tilgung abgezogen und davon die Zinsen berechnet (*Bild 2*).

```
100 CR$=CHR$(13):SK$=CHR$(29)
110 BR$=CHR$(1)
120 PRINT"■":PRINT:PRINT:PRINT
130 PRINTTAB(10)"RATENTILGUNG"
140 PRINT:PRINT
150 INPUT"KAPITAL.......:";K
160 INPUT"JAHRE.........:";J
170 INPUT"ZINSSATZ.....%:";Z
180 OPEN7,4
190 PRINT#7,CR$;BR$;
200 PRINT#7,"RATENTILGUNG";
210 PRINT#7,SK$;CR$;
220 PRINT#7,"KAPITAL....:";K
230 PRINT#7,"JAHRE......:";J
240 PRINT#7,"ZINSSATZ..%:";Z
250 CLOSE7
260 TR=K/J
270 PRINT"J.";TAB(4)"J.BEL.";
280 PRINTTAB(14)"ZI 1/1";
290 PRINTTAB(22)"TILG";
300 PRINTTAB(30)"MON.BEL."
310 OPEN5,4,0
320 PRINT#5,"J.    ";
330 PRINT#5," J.BELASTUNG ";
340 PRINT#5," ZINS JAEHRL.";
350 PRINT#5,"   TILGUNG   ";
360 PRINT#5," MON.BELAST. "
370 CLOSE5
380 OPEN6,4,2
390 OPEN5,4,1
400 PRINT#6,"ZZ  ";
410 PRINT#6," 99999999.99 ";
420 PRINT#6," 99999999.99 ";
430 PRINT#6," 99999999.99 ";
440 PRINT#6," 99999999.99 "
450 FORI=1TOJ
460 REM JAEHRL. RESTSCHULD
470 TT=(K-TR*(I-1))
480 REM JAEHRLICHER ZINS
490 ZT=TT*Z/100:C=TR+ZT
500 U=INT(1E+2*C+.5)*1E-2
510 V=INT(1E+2*ZT+.5)*1E-2
520 W=INT(1E+2*TR+.5)*1E-2
530 X=INT(1E+2*(C/12))*1E-2
540 PRINTI;TAB(3)U;TAB(14)V;
550 PRINTTAB(22)W;TAB(30)X
560 PRINT#5,I;U;V;W;X
570 REM ENDSUMME ZINS
580 R=R+ZT
590 REM GESAMTAUFWAND
600 GA=R+K:J=J-1:NEXTI:PRINT
610 CLOSE6:CLOSE5
620 PRINT"ZINS          :"R
630 PRINT"GESAMTAUFWAND:"GA
640 OPEN9,4
650 PRINT#9,CR$;
660 PRINT#9,"GESAMTBELASTUNG:"GA
670 PRINT#9,"ZINSBELASTUNG  :"R
680 CLOSE9
```

Bild 1. Programm für die Mikrocomputer PET und CBM. Zu beachten ist, daß die formatierten Druckbefehle (z. B. Zeile 410) nur mit dem „intelligenten" Commodore-Drucker funktionieren

Ohne Drucker können folgende Zeilen weggelassen werden: 100, 110, 180...250, 310...440, 560, 610, 640 bis 680. Das Steuerzeichen „invertiertes Herz" in Zeile 120 bedeutet Clr = Bildschirm löschen. Dr. Fritz Galland

```
RATENTILGUNG
KAPITAL....: 1000
JAHRE......: 10
ZINSSATZ..%: 10
J.    J.BELASTUNG  ZINS JAEHRL.   TILGUNG    MON.BELAST.
01        200.00        100.00      100.00         16.66
02        190.00         90.00      100.00         15.83
03        180.00         80.00      100.00         15.00
04        170.00         70.00      100.00         14.16
05        160.00         60.00      100.00         13.33
06        150.00         50.00      100.00         12.50
07        140.00         40.00      100.00         11.66
08        130.00         30.00      100.00         10.83
09        120.00         20.00      100.00         10.00
10        110.00         10.00      100.00          9.16

GESAMTBELASTUNG: 1550
ZINSBELASTUNG  : 550
```

Bild 2. Ausdruckbeispiel für die Ratentilgung. Wie man sieht, rundet der Drucker auf zwei Stellen nach dem Komma

Eine PROM-Weiche für den AIM-65

Der Sockel Z24 (Adr: D000...DFFF) wird beim AIM-65 normalerweise für den Assembler benutzt. Diejenigen Leser, die ausschließlich in Basic programmieren, können an dieser Stelle die Basic-Erweiterung von GWK einsetzen. Der Autor benutzt jedoch sowohl Basic als auch Assembler. Anfangs glaubt man, durch einfaches Austauschen der ROMs ließe sich gut arbeiten. Ganz abgesehen vom ständigen Öffnen und Schließen des Gehäuses werden aber die Beinchen der ICs müde und drohen abzubrechen.

Man müßte beide ROMs gleichzeitig und am gleichen Platz einsetzen können. Dieser Gedanke führte zu folgender Lösung: Auf einer kleinen Platine von 70 mm x 35 mm (*Bild 1*) befinden sich drei Sockel mit paralleler Verdrahtung. Während am Sockel 1 und 3 die Beinchen der Sockel verkürzt werden, wird an die Beinchen des mittleren Sockels eine DIL-Steckerplatte angelötet, jedoch mit Ausnahme des Pin 20, der am Sockel verkürzt und an der Steckerplatte nach außen weggebogen wird. Von der DIL-Steckerplatte wird von Pin 20 eine Chip-Select-Leitung und von Pin 24 eine +5-V-Leitung an einen Schalter gelegt. Von hier aus laufen drei Leitungen wieder an die Platine zurück und werden an die mit „1, 2, 3" gekennzeichneten Stellen angelötet. Bei Benutzung der Platine mit drei PROMs wird ein 3 x 3-Schalter benötigt, der nach der Wahrheitstabelle in *Bild 2* zu verdrahten ist. Beim Autor sind jedoch z. Z. nur zwei Sockel belegt, wodurch sich die Schaltung vereinfacht und ein doppelpoliger Miniaturumschalter ausreicht. Seine Verdrahtung geht aus *Bild 3* hervor.

Reiner Kluth

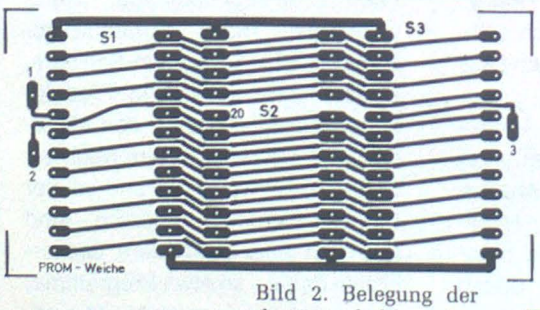


Bild 1. Platine für maximal drei PROMs. Damit ist es möglich, beim AIM-65 wahlweise den Assembler, die GWK-Basic-Erweiterung oder ein selbstbeschriebenes PROM im Adressenbereich D000...DFFF zu betreiben

	Schalter a	b	c
Sockel 1	$\overline{cs}$	1	1
Sockel 2	1	$\overline{cs}$	1
Sockel 3	1	1	$\overline{cs}$

Bild 2. Belegung der drei Umschaltkontakte eines 3 × 3-Drehschalters. Verwendet man nur zwei PROMs, so genügt natürlich ein doppelpoliger Umschalter

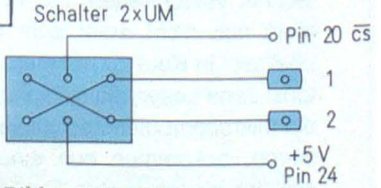


Bild 3. Verdrahtung des doppelpoligen Umschalters für zwei PROMs

Ing. grad. Paul Hlubek

Bausparen mit Basic

Das im folgenden beschriebene und in Bild 1 aufgelistete Basic-Programm in der PET-Microsoft-Syntax gibt einen Kontoauszug wahlweise auf dem Bildschirm oder auf dem CBM-Drucker aus. Dabei wird mit vier Bewertungs-Stichtagen pro Jahr gerechnet (leider trifft dies nicht für alle Bausparkassen zu).

Der „Witz" des Programms ist, daß auch rückwärts gerechnet werden kann: Man kann fiktive Einzahlungen eingeben und sie dann Schritt für Schritt wieder rückgängig machen. Diese Technik erlaubt das für den Vertrag sehr wichtige „Timing". Ein zu früh oder zu spät zugeteilter Bausparvertrag kostet nämlich Geld, das man sich hätte sparen können.

Da – im Gegensatz zur Realität – nach jeder einzelnen Zahlung ein aktueller Kontoauszug (*Bild 2*) erstellt wird, erhält man die nötige Transparenz, um seine Bauspardaten in gewünschter Weise zu steuern.

Eines kann das Programm natürlich nicht: Es weiß nicht, welche Bewertungszahl gerade zur Zuteilung genügt. Diese Information läßt sich aber jederzeit leicht beim zuständigen Gebietsvertreter einholen. Da diese Zahl in absehbarer Zeit keinen großen Schwankungen unterliegt, lassen sich durchaus realistische Prognosen treffen.

Wer etwas Basic-Erfahrung hat, kann sich das Programm leicht für seinen eigenen Bedarf zurechtschneiden. Ein alter Kontoauszug liefert dabei die Kontrolle; Differenzen von einigen wenigen Pfennigen können dabei durchaus auftreten. Die Bewertungszahl errechnet sich wie folgt: alte Bewertungszahl plus Verhältnis von Kontostand am Bewertungsstichtag zur Bausparsumme. Die Verzinsung ist vierteljährlich bzw. monatlich bei Einzahlungen über 1000 DM.

Das Herz des Programms befindet sich in Zeile 2300. Hier wird geprüft, ob mit der letzten Einzahlung ein neuer Bewertungsstichtag überschritten wurde. Bei negativem Ergebnis wird die Einzahlung zwischengespeichert, anderenfalls werden Kontostand, Zinsen, Bewertungszahl, Guthaben (in Prozent der Bausparsumme) sowie das neueste Datum errechnet und angezeigt.

Der Bediener hat nun folgende Möglichkeiten der Eingabe: Er kann weitere Einzahlungen tätigen; bei Mißfallen des Vertrages können die letzten Einzahlungen wieder gelöscht werden; die neuesten Daten können in den Programmspeicher gebracht werden.

Letzteres hat den Vorteil, daß der neueste Stand des Kontos auf Band gespeichert werden kann. Parallel zu den tatsächlichen Zahlungen bleibt das Programm also „up to date". Selbstverständlich darf man auch negative Beträge einzahlen, z. B. anfallende Gebühren.

Mehrere Einzahlungen innerhalb eines Monats werden zusammengefaßt. Insgesamt sind acht Einzahlungen pro Jahr zugelassen. Die letzten vier Zahlungen erscheinen auf dem Bildschirm. Dies mag auf den ersten Blick wenig erscheinen, da doch das Jahr zwölf Monate hat. Aus folgender Überlegung ist es jedoch ungeschickt, monatlich zu zahlen: Ein normales Sparkonto bringt wesentlich mehr Zinsen als das Bausparkonto. Wesentlich ist

Tabelle:
Belegung der Variablennamen

VB$	Vertragsbeginn
BS	Bausparsumme
TA$	Guthabenzins
KS$	Alter Kontostand
BZ	Bewertungszahl
KZ(C)	Teilzinsen für vollendete Quartale
K1(O)	Monat } alter Kontostand
K2(O)	Jahr } alter Kontostand
K3(O)	Betrag } alter Kontostand
D$(O)	Monat/Jahr } alter Kontostand
TA	Guthabenzinszahl
FF	Korrekturfaktor für „Integer"
C	Anzahl der Einzahlungen
KM(C)	Einzahlmonat
KJ(C)	Einzahljahr
KB(C)	Einzahlbetrag
SP(O)	Guthaben in % der Bausparsumme
S1	Anfangsadresse von Bauspardaten im Programmspeicher
S2	Endadresse von Bauspardaten im Programmspeicher
HH$	Hilfsvariable für POKE
HL$	dito
HR$	dito
L$	CHR$(157)
W	Laufindex für Einzahlungen
KZ(C)	Zinsen
KB(C)	Endbetrag
L(W)	Lastschrift
K5	Einzahlungsbetrag über 1000 DM
K6	Einzahlungsmonat über 1000 DM
P	Flag für Beträge über 1000 DM
L	Flag für die Anzahl der Monate mit über 1000 DM
TZ	Teilzinsen
KK(C)	Zwischenspeicher für Einzahlungen im nächsten Bewertungsmonat
P$	CHR$(17)
F1$	Formatierungsvorschrift
F2$	Formatierungsvorschrift
S$	Skip
V$	„|"
EE$	„Einzahlung" + „W"

KONTOAUSZUG

SPARSUMME (% BS-S.)	BEGINN	BEW.-ZAHL	BS-SUMME
.00	12/79	.0000	100000.00

BUCHUNG	WERT	LASTSCHR.	GUTSCHR.
KONTOSTAND	12/79	.00	.00
ZINSEN	12/79	.00	.00
KONTOSTAND	12/79	.00	.00

Bild 2. Formschön und übersichtlich erstellt das Programm aktuelle Kontoauszüge

```
0 GOTO100
1 VB$="12/79"
2 BS=100000
3 TA$="2.5%"
4 KS$="12/79/000000000"
5 BZ=0000000
6 KZ(C)=00000000
10 K1(0)=VAL(LEFT$(KS$,2)) : K2(0)=VAL(MID$(KS$,4,2))
20 K3(0)=VAL(MID$(KS$,7))
30 K$="KONTOSTAND" : E$="ZAHLUNG" : Z$="ZINSEN"
40 D$(0)=STR$(K1(0))+"/"+RIGHT$(STR$(K2(0)),2)
50 TA=VAL(TA$) : FF=1+1E-8
60 KM(C)=K1(0):KJ(C)=K2(0):KB(C)=K3(0):K4(C)=INT(KB(C))
70 DK$(C)=D$(0):BZ(0)=BZ
80 SP(0)=INT(KB(C)*10000/BS)/100
90 GOTO1000
100 PRINTCHR$(147)"NEUER VERTRAG ?J/N?"
110 WAIT525,1:GETNV$:IF NV$="N"THEN1
500 PRINT"■SETZEN DER BAUSPARDATEN":PRINT
520 PRINT"VERTRAGSBEGINN [NUR MONATE 03,06,09,12 ANGEBEN] Z.B. 03/75 ";
530 INPUT VB$ : IF LEN(VB$)<>5 THEN PRINT"■??■";:GOTO530
540 FOR I=1043 TO 1047 : POKEI,ASC(MID$(VB$,I-1042,1)): NEXT I
550 PRINT:PRINT"BAUSPARSUMME= [Z.B.  050000  ]"
560 INPUT BS$ : IF LEN(BS$)<>6 THEN PRINT"■??■";:GOTO560
570 FOR I=1057 TO 1062 : POKEI,ASC(MID$(BS$,I-1056,1)): NEXT I
580 PRINT:PRINT"VERZINSUNG (IN %) [ Z.B. 3.0 ]
590 INPUT TA$ : IF LEN(TA$)<>3 THEN PRINT"■??■";:GOTO590
600 FOR I=1073 TO 1075 : POKEI,ASC(MID$(TA$,I-1072,1)) : NEXT I
610 PRINT:PRINT"KONTOSTAND AM ENDE EINES QUARTALS =     Z.B. 09/77/1504.42"
620 INPUT"    ";KS$ : DK$(C)=LEFT$(KS$,5) : KB(C)=VAL(MID$(KS$,7))
660 GOSUB 9000
680 S1=1102:S2=1094:GOSUB 9060
700 PRINT:PRINT"BEWERTUNGSZAHL= [ Z.B.  1.4752  ]"
720 INPUT BZ(C)
760 S1=1118 : S2=1112 : GOSUB 9080
800 PRINT:PRINT"TEILZINSSUMME FUER VOLLENDETE QUARTALE =  [ Z.B. 144.22  ]"
820 INPUT KZ(C)
840 S1=1137 : S2=1130 : HH$=LEFT$(STR$(KZ(C)),9) : GOSUB 9100
900 GOTO 1
1000 PRINTCHR$(147)SPC(8)"KONTOAUSZUG      "D$:L$=CHR$(157)
1020 FORI=1TO40:PRINT"*";:NEXT:PRINT
1040 PRINT"■SPARSUMME■ ";"  ■BEGINN■  ■BEW.-ZAHL■ ■BS-SUMME■"
1050 GOSUB12000
1060 PRINTSP(C)" %";TAB(12)" ";VB$,BZ(C),BS
1070 GOSUB12000
1080 PRINT" ■BUCHUNG■","  ■WERT■     ■LASTSCHR. GUTSCHR."
1090 GOSUB12000
1100 PRINTK$;"  "D$(0),L(0)"       "K3(0)L$
1120 GOSUB 12000
1200 W=C-3 : GOSUB 10000
1250 W=C-2 : GOSUB 10000
1300 W=C-1 : GOSUB 10000
1450 W=C : GOSUB 10000
1600 PRINTZ$;"        "DK$(C),"     " ,KZ(C) L$
1620 GOSUB12000
1640 IF KB(C)<0 THEN LB(C)=KB(C):GOTO1700
1660 HB(C)=KB(C)
1700 PRINTK$;"  "DK$(C),LB(C),HB(C)L$L$
1720 GOSUB12000
1800 IFDD=1THEN6400
2000 GOSUB15000:PRINT"WEITERE EINZAHLUNGEN?";
2030 GETWE$:IFWE$=""THEN2030
2040 IFWE$="N"ORC=8THEN5000
2050 C=C+1:INPUT"?MONAT/JAHR/BETRAG?";K$(C)
2060 K1(C)=VAL(LEFT$(K$(C),2)):K2(C)=VAL(MID$(K$(C),4,2))
2070 K3(C)=VAL(MID$(K$(C),7))
2080 K3=K3(C) : K1=K1(C) : GOTO 11000
2100 D$(C)=STR$(K1(C))+"/"+RIGHT$(STR$(K2(C)),2)
2150 IFK3>=1000ANDK1<3ANDK1<6ANDK1<9ANDK1<12THENK5=K3:K6=K1(C):P=1
2160 IFP=1ANDK1=1ORK1=4ORK1=7ORK1=10THENL=2
2170 IFP=1ANDL<>2THENL=1
2200 BZ(C)=BZ(C-1):KZ(C)=KZ(C-1):K4(C)=K4(C-1):KK(C)=KK(C-1):DZ$(C)=DZ$(C-1)
2210 DK$(C)=DK$(C-1):KB(C)=KB(C-1):KM(C)=KM(C-1):KJ(C)=KJ(C-1):KB(C)=KB(C-1)
2215 TZ(C)=TZ(C-1):SP(C)=SP(C-1)
2220 DK$(C)=RIGHT$(STR$(KM(C)),2)+"/"+RIGHT$(STR$(KJ(C)),2)
2300 KM(C)=KM(C)+3
2320 IFKM(C)>12THENKM(C)=KM(C)-12:KJ(C)=KJ(C)+1
2340 IFKM(C)>K1(C)ANDKJ(C)>=K2(C)ORKJ(C)>K2(C)THEN2400
2360 GOTO2500
2400 KM(C)=KM(C)-3
2420 IFKM(C)<1THENKM(C)=KM(C)+12:KJ(C)=KJ(C)-1
2440 KK(C)=KK(C)+K3
2460 GOTO1000
2500 Z(C)=INT((K4(C)*TA/4)*FF)/100
2510 IFK1<=KM(C)ANDK2(C)<=KJ(C)THENTZ(C)=INT((P*K5*L*TA/12)*FF)/100
2530 IFKM(C)=3THENKZ(C)=0
2540 KZ(C)=KZ(C)+Z(C)+TZ(C)
2550 TZ(C)=0:P=0
2560 IFKM(C)=12THENKB(C)=KB(C)+KZ(C)
2600 IFKM(C)>=K1(C)ANDKJ(C)>=K2(C)THENKB(C)=KB(C)+K3+KK(C):KK(C)=0:K3=0
2610 KB(C)=KB(C)+KK(C):KK(C)=0
2620 K4(C)=INT((KB(C))*FF)
2700 B=INT((K4(C)/BS*1014)*FF)/1014
2720 BZ(C)=BZ(C)+B
2740 SP(C)=INT((KB(C)*10000/BS)*FF)/100
2800 GOTO2220
5000 GOSUB15000:PRINT"EINZAHLUNG GUELTIG ?";
5020 GETEG$:IFEG$=""THEN5020
5040 IFEG$="J"THEN6000
5050 IFC=0THEN2000
```

```
5060 L(C)=0:H3(C)=0
5100 C=C-1
5950 GOTO1000
6000 DK$(C)=RIGHT$(DK$(C),5) : GOSUB 9000
6020 S1=1102 : S2=1094 : GOSUB 9060
6040 S1=1118 : S2=1112 : GOSUB 9080
6060 S1=1137 : S2=1130
6080 HH$=LEFT$(STR$(KZ(C)),9)
6100 GOSUB 9100
6200 GOSUB15000 : PRINT"SOLL KONTOAUSZUG GEDRUCKT WERDEN ?";
6220 GET KD$ : IF KD$="" THEN 6220
6240 IF KD$="J" THEN 20000
6900 END
9000 DD$(C)=DK$(C)+"/"
9040 RETURN
9060 HH$=LEFT$(STR$(KB(C)),10) : GOTO 9100
9080 HH$=LEFT$(STR$(BZ(C)),8)
9100 FOR I=S1 TO S2 STEP -1
9140 HR$=RIGHT$(HH$,ABS(I-S1-1))
9160 HL$=LEFT$(HR$,1)
9180 IF HL$=" " THEN HL$="0"
9200 POKE I,ASC(HL$) : NEXT I
9220 RETURN
10000 IF W<=0 THEN PRINT L$ : GOTO 10400
10100 IF K3(W)<0 THEN L(W)=K3(W) : GOTO 10300
10200 H3(W)=K3(W)
10300 PRINT E$W;"  "D$(W),L(W),H3(W);L$
10400 GOSUB 12000
10500 RETURN
11000 PRINTCHR$(17);:GOSUB15000:PRINT"WEITERE EINZAHLUNG IN DIESEM MONAT ?",
11100 GETWZ$:IFWZ$=""THEN11100
11200 IFWZ$="N"THEN2100
11300 INPUT"BETRAG";W$
11400 K3(C)=K3(C)+VAL(W$):GOTO2080
12000 PRINTCHR$(145)TAB(11)"|"TAB(19)"|"TAB(29)"|"
12100 GOSUB12900
12600 PRINT"——+";:GOSUB12900:PRINT"+";
12620 GOSUB12900:PRINT"——+";:GOSUB12900
12700 PRINT"——":PRINTCHR$(145);:RETURN
12900 FORI=1TO7:PRINT"-";:NEXT:RETURN
15000 PRINTCHR$(19);:P$=CHR$(17)
15020 PRINTP$P$P$P$P$P$P$P$P$P$P$P$P$P$P$P$P$P$P$P$P$P$P$;
15040 FOR I=1 TO 40 : PRINT" ";:NEXT
15060 RETURN
20000 OPEN1,4:OPEN3,4,1
20020 OPEN2,4,2
20100 PRINT#1,,CHR$(1),,"KONTOAUSZUG"
20200 PRINT#1,,,:FOR I=1TO 52 : PRINT#1,CHR$(36);:NEXT I : PRINT#1:PRINT#1
20300 PRINT#1,,,"SPARSUMME (% BS-S.)| BEGINN   | BEW.-ZAHL | BS-SUMME"
20400 GOSUB26000
20500 F1$="                    999.99      A  AAAAA   A99999.9999A999999.99"
20600 PRINT#2,F1$
20650 S$=CHR$(29) : V$="|"
20700 PRINT#3,SP(C);V$S$VB$S$V$S$BZ(C);V$S$BS
20800 GOSUB 26000
20900 PRINT#1,,,"BUCHUNG                | WERT      | LASTSCHR.| GUTSCHR."
20940 F2$="          AAAAAAAAAAAA        A  AAAAA   A 999999.99A999999.99"
20960 PRINT#2,F2$
20980 GOSUB 26000
21000 PRINT#3,K$S$V$S$D$(0)S$V$S$L(0);V$S$K3(0);CHR$(13);
21010 GOSUB 25000
21020 W=C-7 : GOSUB 30000
21040 W=C-6 : GOSUB 30000
21060 W=C-5 : GOSUB 30000
21080 W=C-4 : GOSUB 30000
21100 W=C-3 : GOSUB 30000
21120 W=C-2 : GOSUB 30000
21140 W=C-1 : GOSUB 30000
21160 W=C   : GOSUB 30000
21300 PRINT#3,Z$S$V$S$DK$(C)S$V$S$;Z;V$S$KZ(C);CHR$(13);
21400 GOSUB25000
21500 PRINT#3,K$S$V$S$DK$(C)S$V$S$;LB(C);V$S$HB(C);CHR$(13);
21600 PRINT#1,,,:FORI=1TO52:PRINT#1,CHR$(35);:NEXT I
23000 PRINT#1,S$,CHR$(13)
23100 PRINT#3,S$,CHR$(13)
23200 PRINT#2
23500 CLOSE1 : CLOSE2 : CLOSE3
24000 END
25000 PRINT#1,,,"————————————————————";
25100 PRINT#1,"+————————+————————+————————":RETURN
26000 PRINT#1,,,"————————————————————————————————————————————————":RETURN
27000 PRINT#1,,,"                    |          |          |"
27100 RETURN
30000 IF W<=0 THEN 31000
30100 IF K3(W)<0 THEN L(W)=K3(W) : GOTO 30300
30200 H3(W)=K3(W)
30300 W$=RIGHT$(STR$(W),1):EE$=CHR$(160):EE$=E$+EE$:EE$=EE$+W$
30400 PRINT#3,EE$S$V$S$D$(W)S$V$S$L(W);V$S$H3(W);CHR$(13);
30500 GOTO31100
31000 GOSUB27000
31100 GOSUB25000
31200 RETURN
31300 END
40000 REM ******************************
40010 REM * PROGRAMMIERER: PAUL HLUBEK*
40020 REM *    ALL RIGHTS RESERVED    *
40030 REM ******************************
READY.
```

Bild 1. Sparen beim Bausparen ermöglicht dieses PET-Basic-Programm

lediglich, rechtzeitig vor dem jeweiligen Bewertungsstichtag sein Schärflein zu entrichten. Am unteren Rand des Bildschirms sind zwei Zeilen reserviert, um mit dem Computer im Dialog arbeiten zu können. Solange die Fragen des Computers nicht beantwortet werden, bleibt der letzte Kontoauszug auf dem Bildschirm erhalten. Will man lediglich testen, wie sich der Vertrag ohne Einzahlungen entwickelt, so gibt man zum Beispiel 12/81/0 (Monat/Jahr/Betrag) ein.

Die zahlreichen Integer-Funktionen im Programm versuchen, den Kontoauszug der Sparkassen möglichst genau zu treffen. Aus verständlichen Gründen runden sie so oft wie möglich ab. Programmtechnisch interessant, wenn auch für die Benutzer anderer Computer nicht ohne weiteres so übernehmbar, ist das direkte Eingreifen in den Programmspeicher, insbesondere bei unformatierter Eingabe, z. B. mit Vornullen. Die Belegung der Variablennamen geht aus der *Tabelle* hervor.

Zufallszahlen mit Z80 und 6502

Ein schneller und für Mikrocomputer gut geeigneter Algorithmus zur Erzeugung von Pseudozufallszahlen ist durch die folgende Beziehung gegeben:

$X_k = RL_n (X_{k-1} + X_{k-2})$

$Y_k = X_k \bmod 2^m$

Das Symbol „+" steht für eine bitweise Exklusiv-Oder-Verknüpfung, RL_n bedeutet eine Linksrotation um n Bit. Y_k ist das Ergebnis mit einer Länge von m Bit. Die Wortlänge l der X_k muß also größer oder gleich m sein. Um eine möglichst geringe Korrelation der Y_k untereinander und eine große Periode der Zufallszahlenfolge zu erzielen, sollten die folgenden Bedingungen erfüllt sein:

```
LOC   OBJ CODE M STMT SOURCE STATEMENT
                                ZUFALL
0000  D9          1  ZUFALL:  EXX
0001  EB          2           EX DE,HL
0002  44          3           LD B,H
0003  7D          4           LD A,L
0004  AB          5           XOR E
0005  67          6           LD H,A
0006  78          7           LD A,B
0007  AA          8           XOR D
0008  CB11        9           RL C
000A  1F         10           RRA
000B  CB19       11           RR C
000D  6F         12           LD L,A
000E  D9         13           EXX
000F  C9         14           RET
```

Bild 1. Z-80-Programm für die Erzeugung von Pseudo-Zufallszahlen. Zur Initialisierung kann hier recht einfach das Refresh-Register dienen – es liefert einen zufälligen Anfangswert

```
0500  86 e9        stx e9
0502  a5 e5        lda e5
0504  aa           tax
0505  45 e7        eor e7
0507  86 e7        stx e7
0509  aa           tax
050a  a5 e6        lda e6
050c  86 e6        stx e6
050e  aa           tax
050f  45 e8        eor e8
0511  86 e8        stx e8
0513  85 e5        sta e5
0515  06 e4        asl e4
0517  66 e6        ror e6
0519  66 e4        ror e4
051b  a6 e9        ldx e9
051d  60           rts
```

Bild 2. Zufallszahlen-Programm für den Mikroprozessor 6502; es arbeitet nach dem gleichen Algorithmus

– l ist möglichst groß zu wählen,
– für n muß gelten: $m \leqq n < l$,
– l darf kein ganzzahliges Vielfaches von n sein.

Die Implementierung wird besonders einfach für l = 16, m = 8 und n = 8. Die dritte der obigen Bedingungen wird erfüllt, indem man die Rotation durch das Carry-Bit erfolgen läßt. Dadurch wird X_k gewissermaßen auf 17 Bit verlängert. Dieses Bit ist natürlich ebenso wie X_{k-1} und X_{k-2} für die folgenden Aufrufe des Zufallszahlenprogramms geeignet abzuspeichern. Die Periode ist $>10^9$, was für die meisten Anwendungen ausreichen dürfte.

Die abgebildeten Unterprogramme für die Prozessoren Z80 (*Bild 1*) und 6502 (*Bild 2*) liefern eine 8-Bit-Zufallszahl im Akkumulator. Im Z80-Programm werden X_{k-1}, X_{k-2} und Carry in den Registern C', D', E', H' und L' des häufig unbenutzten Zweitregistersatzes abgespeichert, in der 6502-Version dienen hierzu fünf Speicherplätze in Page Null.

Eine Initialisierung der Zufallszahlenfolge (manche Basic-Interpreter kennen hierzu den Befehl RANDOMIZE) ist beim Z80 sehr einfach: Man lädt eines der Zweitregister H', L', D' oder E' mit einem Wert auf dem Refresh-Register R.

Klaus Betke

Repeat-Funktion beim CBM 3000

Die Umschaltung aller Tasten auf Dauerfunktion (Repeat-Taste) erleichtert die Arbeit mit dem CBM 3000 ganz erheblich. Um dies zu erreichen, wird der Interrupt-Vektor auf das Programm (*Bild*) im 2. Kassetten-Buffer (03D5) gesetzt. Durch Benutzung der EOR-Funktion kann mit SYS 966 die Repeat-Funktion eingeschaltet und durch erneutes SYS 966 wieder ausgeschaltet werden. Nach dem Einschalten wird dann das Programm bei jedem Interrupt, also 60mal in der Sekunde, ab 03D5 durchlaufen. Wird nun irgendeine Taste gedrückt, so erscheint sofort das Zeichen auf dem Bildschirm. Danach wartet das Programm ca. 0,3 Sekunden, ehe dann fortlaufend mit 15 Zeichen pro Sekunde geschrieben wird. Die Wartezeit ist notwendig, um auch die sichere Eingabe eines Zeichens zu gewährleisten. Durch Verändern der Speicherzelle 03E2 (dez. 994) kann die Wartezeit variiert werden. Soll die Wiederholrate verändert werden, ist dies in 03f5 (dez. 1013) möglich. Wichtig ist, daß der Cursor bei der schnellen Bewegung sichtbar bleibt. Dies geschieht mit den Befehlen 03F0 und 03F2.

```
REPEAT - TASTE

03C6  78           SEI
03C7  A5 90        LDA 0090   UMSCHALTUNG
03C9  49 FB        EOR FB
03CB  85 90        STA 0090   DES IRQ -
03CD  A5 91        LDA 0091
03CF  49 E5        EOR E5     VEKTORS
03D1  85 91        STA 0091
03D3  58           CLI
03D4  60           RTS
03D5  A5 97        LDA 0097   WIRD EINE
03D7  C9 FF        CMP FF
03D9  F0 04        BEQ 03DF   TASTE
03DB  C5 B7        CMP 00B7
03DD  F0 09        BEQ 03E8   GEHALTEN ?
03DF  85 B7        STA 00B7
03E1  A9 14        LDA 14     WARTEZAHLER
03E3  85 B8        STA 00B8
03E5  4C 2E E6     JMP E62E   EXIT
03E8  C6 B8        DEC 00B8
03EA  D0 F9        BNE 03E5   WARTEN !
03EC  A9 FF        LDA FF
03EE  85 97        STA 0097   REPEAT
03F0  A9 02        LDA 02
03F2  85 A8        STA 00A8   CURSOR HELL
03F4  A9 04        LDA 04
03F6  D0 EB        BNE 03E2   EXIT
```

Durch ein kleines Interrupt-Programm im Kassetten-Buffer 2 kann man eine Dauerfunktion aller Tasten realisieren, was die Eingabe manchmal sehr erleichtert

Wie alle Programme, die den Interrupt-Vektor verstellen, muß auch dieses vor Aufnahme oder Wiedergabe mit der Kassetteneinheit ausgeschaltet werden.

Christian Bentert